A Comprehensive Full-Scale Digital Twin Model for Enhancing Airport Security Checkpoint Efficiency

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

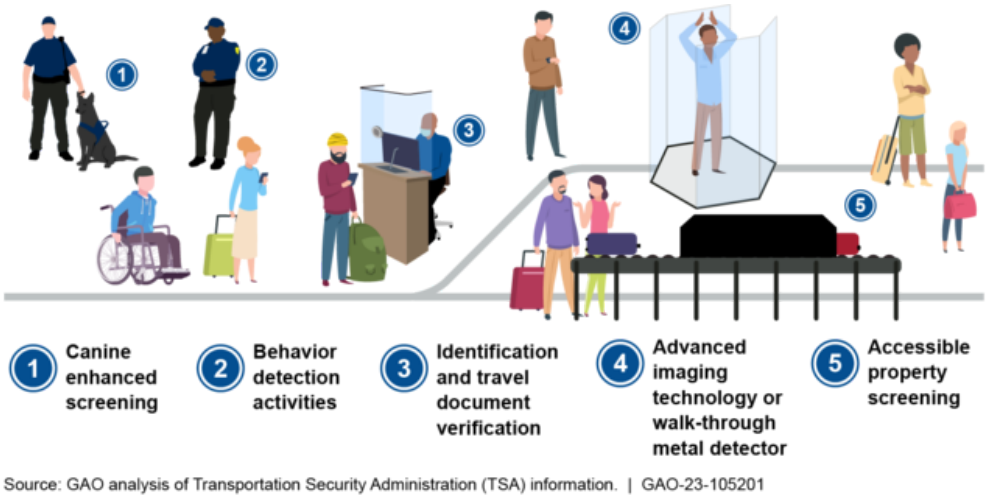
Carnegie Mellon University

Abstract

This comprehensive report explores the integration of advanced technological solutions to optimize the Transportation Security Administration (TSA)'s operations, particularly focusing on enhancing the security screening process at national airports. Established under the Aviation and Transportation Security Act of 2001, the TSA aims to secure national transportation systems while ensuring freedom of movement for people and commerce. Recognizing the critical balance between security needs and operational efficiency, this paper reviews the TSA security check process and its challenges, particularly in the context of queue management and passenger throughput.The research delves into the applicability of Digital Twin technology in modeling TSA queue dynamics and capacity analysis. We explore how this technology can be utilized to forecast capacities and manage passenger flow more effectively, contributing to the overall satisfaction of travelers and the timeliness of flights. In tandem with this, we investigate the role of Google's latest Large Language Model (LLM), PaLM 2, and its associated MakerSuite platform. This exploratory project examines how these advanced computational tools can assist in transforming Operations Management models into machine-readable formats, enhancing the Digital Twin framework. Two specific use cases of the PaLM are considered: codifying existing Operations Management models for automation and developing a user-friendly platform for end-users to interact with the Digital Twin system, thereby making informed decisions based on the underlying queue management processes.Building on our previous work that established the feasibility of conceptualizing the airport screening process within a Digital Twin framework, we now present a prototypical full-stack Digital Twin implementation. This implementation includes a fully functioning front-end and back-end centered on a question-answering (QA) system, designed to improve human-computer interaction and ease of use. By merging theoretical perspectives from Operations Management with practical applications of generative AI and Digital Twin technologies, this report aims to add to the literature on optimizing TSA operations and highlights new directions for research in domain-specific applications of advanced computational models.

***Important Notice**: This document serves as an academic course project and should not be construted as an authoritative analysis of real-world TSA operations.*

I. Introduction: A Dive into the TSA Process and Our Proposed Solution



The Transportation Security Administration (TSA) plays an essential role in safeguarding air travel within the United States. At the core of its responsibilities is a thorough security screening process, involving steps such as ID verification, baggage and AIT (Advanced Imaging Technology) X-rays, and, when needed, physical pat-downs. Although crucial for maintaining security, the time taken for these processes can vary greatly between checkpoints, often resulting in bottlenecks and operational inconsistencies. Such variations not only interrupt the smooth running of airport operations but also lead to wider issues, including flight delays, financial burdens for airlines due to rescheduling, and reduced passenger satisfaction.

At TSA checkpoints, which are vital for maintaining airport security and managing passenger movement, inefficiencies often cause frustration among both airlines and travelers. The time it takes to process passengers can differ significantly at each checkpoint, impacting the effectiveness of TSA agents and leading to system-wide delays. These issues are exacerbated by improper staffing, with some areas being overstaffed and others understaffed, increasing the likelihood of delays and missed flights. Such scenarios result in immediate inconvenience and have broader financial implications for airlines, while also negatively affecting customer satisfaction.

In light of the continuous growth in air travel, there is an increasing need to improve these processes for enhanced efficiency. Our proposed solution involves the integration of a Digital Twin model to boost the efficiency of the TSA screening process. This innovative model aims to meticulously analyze and refine passenger flow, focusing on reducing bottlenecks to improve operational performance.

The implementation of this model is expected to bring several benefits:

- **Enhanced Traveler Planning**: This will be achieved through precise predictions of waiting times.
- **Strategic Personnel Allocation**: The model will enable the wise distribution of additional staff during peak travel times.
- **Increased Passenger Satisfaction**: A smoother and more efficient process is anticipated to enhance the overall experience for travelers.
- **Timely Departures:** Airlines are expected to benefit from more timely departures, reducing costs associated with delayed takeoffs and disruptions in scheduled air traffic.

The adoption of Digital Twin technology in TSA operations signifies a major move towards modernization, focusing on improving security measures and the overall travel experience. This initiative is a testament to the commitment to evolving TSA operations, adapting them to meet the rising demands of air travel while prioritizing safety, efficiency, and passenger satisfaction. Moreover, the potential applications of this model extend beyond airport security, including crowd management at large events like concerts and improving immigration processes at international airports.

II. An In-Depth Examination of Current Challenges Faced by the TSA

An easily imaginable instance of dysfunction at TSA checkpoints is one where officers temporarily close one component of the screening process to passengers to manage congestion. For example, the initial ID checkpoint is temporarily shut when lines for pat-down scanners or baggage scanners become unmanageable. The main consequence of this challenge is that passengers miss their flights, or that flight operators run on delayed schedules because of long waiting lines, especially during times of peak demand on TSA resources, and TSA workers not being placed optimally in complex and crowded operational structures.

An analysis of passenger travel patterns reveals that this is a significant issue because between 2-8% of passengers miss their flights annually (Elliott, 2018). Given that over 2.8 million individuals navigate through more than 430 airports nationwide daily (Pekoske, 2022), this percentage translates to a notable volume of missed flights. This data implies that there may be discernible reasons contributing to instances of passengers not making their scheduled departures.

As the annual passenger count continues to rise (Passenger Demand Recovery Continues in August, 2023), it becomes imperative for the TSA to enhance its

procedures and guarantee the efficient handling of queues. Moreover, the TSA must factor in unforeseen yet essential delays that may occur during the screening process. Inexperienced travelers, system malfunctions, and security breaches can contribute to additional delays.

We believe that a smart model with Digital Twins can be implemented to alleviate the missed-flights and delayed-flights problem (though this may not directly address the factors that arise from poor passenger planning). Specifically, the Digital Twin has the potential to aid TSA officers in making optimal decisions instead of relying on instinctive judgments when managing checkpoints or shuffling personnel assignments between those checkpoints.

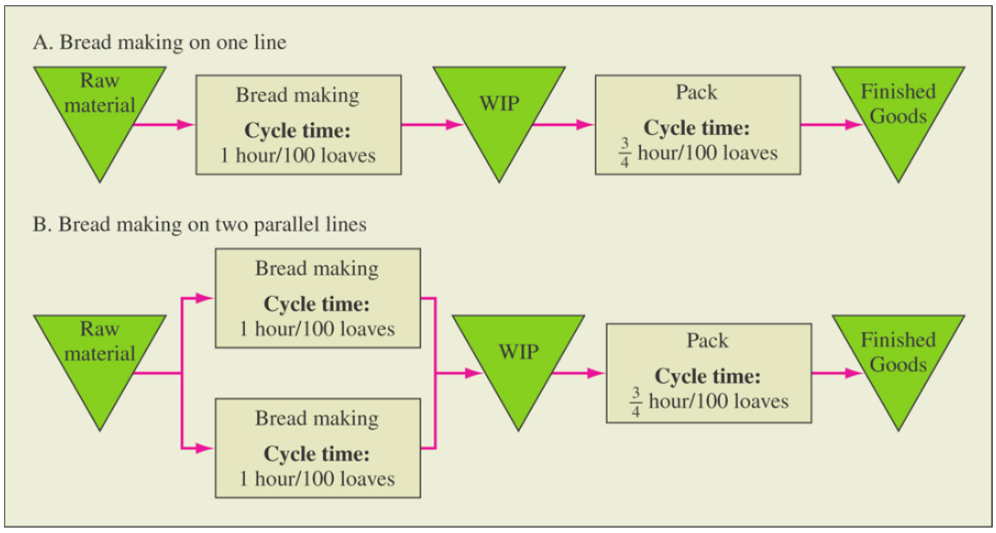
III. Passenger Flow and Capacity Analysis of TSA screening processes

The Operations Management literature has provided ample discourse and analysis on the concepts of line balancing, capacity analysis, and supply chain management, and they have frequently been used to analyze airport passenger flow processes, including but not limited to the security screening process (see citations in abstract), the baggage handling process (Frey et al., 2017), and the check-in and ticketing process (Park & Ahn, 2006).

We will provide a brief exposition of the typical elements of process flow management. This exposition is attributed in its entirety to Stevenson (2020):

Operations Process Flow Management

First, the entire process is visualized as a diagram with flow units (e.g. products, customers) waiting to be processed by resources (processing units) in a given sequence. This is represented in the figure below (*Evaluating Process Capacity*, n.d.) where resources (ovens and packing machines) are used to conduct a certain processing activity (baking and packing) on the given flow unit (bread) in a fixed sequence:



Given this framework, the overall capacity of the system can be analyzed, and it will be defined as the maximum number of flow units that can be processed by a system that is in steady state (Stevenson, 2020).

Notably, capacity can be analyzed on two levels, the first being resource-level capacity (at each individual resource) and process-level capacity (which is the capacity of the entire process and thus equals the capacity of the lowest-capacity, bottleneck resource in the line) (Stevenson, 2020).

Key metrics include:

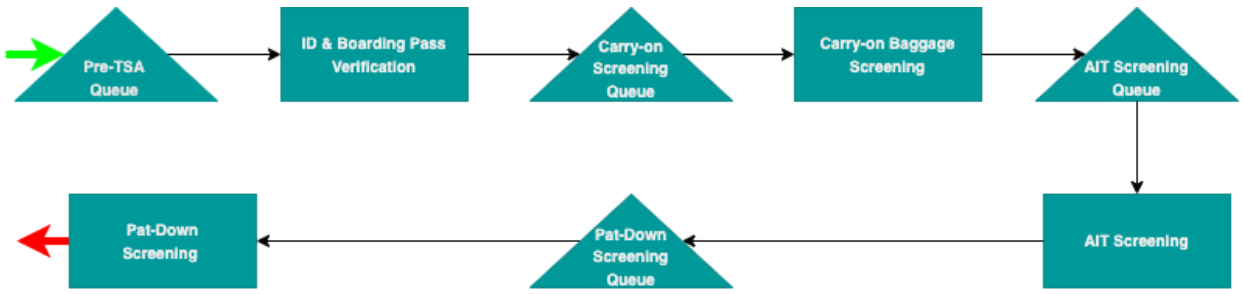
1. Activity Time (the time needed by each individual resource to complete its task)
2. Resource Capacity (1 / Activity Time)
3. Process Capacity (equals the lowest resource capacity)
4. Stage Cycle Time (the aggregate capacity of a stage where there are multiple parallel resources e.g. multiple ovens for baking, or multiple X-ray screeners)
5. Work-in-process: (the number of units being contained within the system while processing is ongoing)
6. Utilization (process capacity/resource capacity)
7. Flow Time (the total time that each unit takes to complete processing)
8. Overall Process Flow Rate (the rate per unit time at which units are being completely processed) (Stevenson, 2020)

Notably, Little's Law provides (Stevenson, 2020) that:

*Work-in-Process = Flow Time * Flow Rate*

Applying the above, we propose that the following flowchart is an appropriate flow process diagram that clearly indicates the applicability of the above process flow concepts. In particular, triangle units (queues) form buffers for flow units (passengers) while they wait to be processed through the particular activities that occur at each

specific resource (rectangles), with the green and red arrows representing their entry and exit from the entire processing line. Notably, we decided to omit the canine screening process as this is not necessarily a standard component of the TSA process.



In the following section, we discuss the applicability of Digital Twin technologies in modeling this process flow. Before proceeding to that section, it would be helpful for us to delineate the Digital Twin-specific metrics we will measure in the specific nomenclature of process flow frameworks:

<u>Possible parameters to be measured under a TSA-specific process flow management framework:</u>

1. Number of travelers each activity station (screening station) is able to accommodate per unit time (stage capacity)
2. Number of passengers present within the system (WIP)
3. Summary statistics of wait time of travelers at each checkpoint, particularly the mean wait time
4. Average time spent on the entire process (including time at queues)
5. A time study of each checkpoint at airport (in terms of flow and cycle time)
6. Stage Utilization (demand-constrained) or Implied Utilization (supply-constrained)
7. Idle time of employees

IV. A characterization of Digital Twin technologies

“While there has been a recent growth of interest in the Digital Twin, a variety of definitions employed across industry and academia remain” (Jones et al., 2020). We propose to adopt Jones’s framework in defining the scope of a Digital Twin project as it is likely the most robust one in the literature to date, having been created after a review of 92 different publications pertaining to Digital Twins.

In particular, we wish to focus on the following most meaningful aspects of a Digital Twin that have been outlined by Jones (2020):

1. Physical Entities and Environments.
2. Virtual Entities and Environments: Computer-generated representations of all relevant physical entities designed to “replicate the state of the physical environment” such as databases and models.
3. Physical-to-Virtual Connection: The ability to record (on a real-time basis) data pertaining to physical attributes and objects and represent them virtually
4. Virtual Processes: Defined as the “computational techniques employed within the virtual-world, e.g. optimisation, prediction, simulation, analysis, integrated multi-physics, multi-scale, probabilistic simulation” (Jones (2020)).
5. Virtual-to-Physical Connection: The ability to realize and effect the conclusions and outcomes of those virtual processes on the physical components within the system.

V. Implementation: Logic Formulation & Methodologies

Considering the framework defined by Jones (2020), we will be following a systematic approach in our implementation. Although our extensive survey of the literature has determined that Wang et al. (2017) were the first to propose the real-time, digital modeling of airport security lines, we believe that we add to the literature by bringing the subject matter under the formal banner of the Digital Twin framework as follows:

1. *Define physical entities and environment*: Identify all components involved in the TSA screening process, which includes passengers, TSA personnel, screening equipment and guidelines, and a map of the TSA screening checkpoints. Document the operational process involved and the typical passenger flow, such as passenger volume and peak times.
2. *Virtual Entities and Environments*: Develop a computer generated model of the TSA screening process. Create a cloud database to store all relevant data.
3. *Physical-to-Virtual Connection*: Implement the use of sensors (Shaw, 2016), such as the ones currently being used at Houston airport to track security wait times (Shaw, 2016). The sensor could potentially record a unique timestamp every time a bluetooth device (refer to Appendix D) passes by a sensor, or track the position of a unique device as it passes through each stage of the process. Create a cloud server to feed the data into our database.
4. *Virtual Processes*: Utilize our optimization model to identify possible bottlenecks and recommend the reallocation of resources depending on passenger volume and worker utility and capacity.
5. *Virtual-to-Physical Connection*: Create a system to provide feedback output based on the calculations performed in the virtual process. This system will

directly notify the TSA personnel responsible for operations if certain checkpoints require additional resources.

A key tenet of the nexus between the physical and virtual process (that undergirds the concept of the Digital Twin) also requires that one continuously monitors the model's performance at each stage in the framework. As such, our proposed framework will also involve a global and continuous cross-checking of the digital twin's forecasts against real-world results at every given time in order to uphold precision.

On a larger timescale, frequent evaluation and monitoring must also be done to understand changes in technology. For example, the optimization and effectiveness of the Digital Twin model may be updated as technologies such as Near-Field Communication and other communication technologies progressively get implemented into consumer devices.

It is also imperative to incorporate the feedback of TSA officers on the Digital Twin Process since they are affected the most by the model and can apply their experience in preventing the Digital Twin from remaining an academic concept with no connection to reality.

Furthermore, our study contributes significantly to the existing body of knowledge by integrating elements of process analysis. In focusing on aspects of process analysis, we particularly emphasize assessing capacity and applying queuing theory to calculate waiting times. Within the domain of operations management, three primary strategies for capacity planning are recognized: Lag Strategy, Lead Strategy, and Match Strategy. The Lag Strategy concentrates on sustaining sufficient resources to fulfill current demand (Simplilearn, 2023). On the other hand, the Lead Strategy proactively plans for future demand, thereby incorporating extra capacity beyond immediate requirements (Simplilearn, 2023). The Match Strategy, meanwhile, synthesizes both approaches by accounting for present demand, future projections, and broader market conditions (Simplilearn, 2023). Our project is designed to support the TSA in adopting the Match Strategy. It is important to acknowledge that in the context of TSA operations, the concept of 'demand' primarily refers to the influx of passengers, as opposed to the conventional notion of 'units' exiting a system in typical business scenarios.

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.

We intend to realize this objective through a meticulously planned four-phase project, which will be elaborated upon in the subsequent sections.

Phase 1: Base Model Conceptualization of Logic and Initial Prototyping

To demonstrate the viability of this project and showcase the potential of our envisioned model, we developed two base models in Microsoft Excel that will serve as the basis for our fully-fledged model.

Since the incorporation and utilization of real-time data are pivotal aspects of any digital twin project, it would be ideal for us to gain access to existing built-in sensors at US airports (such as the one in Houston) or to otherwise obtain data on passenger locations with a high degree of granularity. However, due to time constraints and budgetary limitations, we opt to construct our foundational model using the publicly accessible TSA Throughput data (*FOIA Electronic Reading Room*, n.d.) that is published online. Specifically, we are most interested in the data for Pittsburgh International Airport.

As a first step, we were able to convert and clean the data to ease data analysis. Second, we assumed a uniform distribution for each arrival process, using that as a proxy for inter-arrival times. We then proceeded to highlight the core assumptions of our model. For this toy model, our assumptions are as follows:

1. Only one employee exists at every stage of the process.
2. Activity times are relatively constant with a variability rating of 2.0.
3. Ignore variabilities/seasonalities arising from differences in the time of the day.

Combining these assumptions with our data, we were able to estimate relevant parameters, including the ones described above:

1. Process Capacity
2. Demand Rate
3. Utilization
4. Service and Arrival Variation
5. Waiting time in each Queue
6. Total number of passengers in TSA
7. Total time from beginning to end
8. Idle time of employees

Model 1: Demand-Constrained Model

The first model that we developed was to replicate a demand-constrained process. Although a TSA process is usually supply-constrained, there are times during a day where the process will be severely deprived of demand. For Pittsburgh International Airport, the data shows that this is usually the case from 19:00pm - 2am.

Model 2: Supply-Constrained Model

The second model that we implemented was to replicate the case of a supply-constrained process. With the supply-constrained process, we had to adjust the stage utilization parameters to reflect implied utilization. Moreover, we had to carefully re-implement our waiting time calculations to reflect a new demand rate. This is due to the fundamental nature of a supply constraint process in which,

Flow rate (Throughput) = min {Process capacity, Demand rate}.

It is essential to realize from this equation that the flow rate of a supply-constrained process will always be equal to the capacity of the most constrained process. Hence, the arrival rate going into the queues will depend on the stage capacity of the stage before.

Limitations of the Current Model:

As base models, the implementation of our models for Phase 1 lack specificity and flexibility. Essentially, our models have several strong assumptions that we must amend in future iterations. For example, our current models are not yet able to take in layout parameters that exist in an actual airport such as the number of Baggage and ID Verification screening lanes that we can have. We have also yet to establish the option of having an alternative checkpoint as opposed to the main entry checkpoint into the airport. Decision-making is also still tedious. Though intuitive for the makers of the model, the outputs may be cryptic for users due to the lack of GUIs. Moreover, we want to precisely answer how many employees and lanes we need at any given period of time. We will address these issues in the second phase of the project.

Phase 2: Advanced Model Development and Integration

Post-Phase 1 Challenges

Numerous obstacles were encountered in the process of refining our baseline implementation during Phase 1, particularly in enhancing our optimization module to more accurately reflect real-world conditions, extending beyond the scope of the initial prototype we had previously completed.

Firstly, integrating nonlinearity into our objective function required us to verify the convexity of the queuing theory function. After establishing this, our attention turned to

ensuring that our constraints met the criteria of a convex program. This step was critical because the effectiveness of our optimization module relies on a convex objective function for valid minimization. Without convex sets, our module would be rendered ineffective.

Another significant challenge was differentiating between global and local minima. Although the convex nature of our objective function implies that any local minima would be the global maxima, we had to be cautious to avoid getting stuck at stationary points during certain iterations of the Gradient Descent Algorithm. This required strategic selection of initial values for decision variables to bypass local minima.

Additionally, the scale of feasible solutions presented a substantial obstacle. With 40 binary decision variables, we faced a total of 2^40 (or 1,099,511,627,776) possible values. Tackling a problem of this scale necessitated considerable computational resources, though it was manageable on a standard personal computer. The Solver had to process over 4000 subproblem iterations to find an optimal solution, underscoring the importance of avoiding errors like Zero Division Errors that could cause system crashes. This necessitated the introduction of additional constraints specific to our domain.

Finally, incorporating multiple parameters into the queuing theory added further complexity, necessitating alterations to nearly all formulas from Mini 1. These changes, however, allowed us to create a unified model applicable to both supply-constrained and demand-constrained scenarios, obviating the need for two distinct models as in our earlier Mini 1 version.

Advance Model Implementation

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.

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.

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. We specify further details pertaining to model mechanics in the following sections.

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, 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.

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). This confirms the reliability of

our model as a cost-effective alternative to manual capacity studies, which are both time-consuming and resource-intensive.

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 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.

Phase 3: Model Conversion to Machine-Readable format via MakerSuite

As stated in the Abstract, we considered two specific use cases for MakerSuite: First, to codify our existing Operations Management models into machine-readable instructions;

second, to develop a platform for end-users of the Digital Twin to harness those machine-readable frameworks to make informed decisions about the underlying queue management process.

This initiative was driven by the expectation that the LLM would be capable of delivering precise wait-time calculations based on predefined input parameters. The integration process involved developing a model that combines the robust data structuring of Excel with the advanced computational abilities of Python and the linguistic proficiency of PALM's LLM.

Application 1: Codifying Operations Management Models into Machine-readable Instructions

In this initial phase of the project, we aimed to push the boundaries of PaLM 2's abilities in recognition, translation, and generation. It is now widely acknowledged that large language models excel at receiving and interpreting language inputs from users, identifying key patterns, and even discerning sentiments. Nevertheless, our objective is to assess whether PaLM can effectively process a four-year accumulation of advanced operations management concepts, necessitating the analysis of not just linguistic inputs, but also mathematical formulas. The success criterion lies in PaLM's capacity not only to extract pertinent user inputs but also to transform our intricate prompt into a functional Python program.

To initiate our analysis, we translated our Phase 1 models into a simplified, human-readable format. Subsequently, we input this transformed version into the text prompt interface within Makersuite, with the request to replicate our Digital Twin model in Python3 code. This endeavor presented a substantial challenge: instructing PaLM to comprehend intricate concepts related to advanced operations capacity analysis, grasp the underlying logic, assimilate knowledge from our model's sample outputs, and transform this understanding into a precise Python program.

The final version of the Python program generated by PaLM can take user inputs in the form of the rate of passenger arrivals, the variability in their arrival times (CVa), the variability in their processing times (CVp), and the duration of employee activities across four critical stages. The output will present an estimate of the expected waiting time for passengers at every stage of the process. Our implementation strategy ensures that these key performance indicators are carefully measured and fed into the system to derive the most accurate wait-time projections for service optimization.

Application 2: Developing a platform for end-users of the Digital Twin

Building on any potential success in Application 1 above, we intend to utilize the Python program generated in the first part of the project as an input for a question-answering system in MakerSuite. Leveraging the Chat Prompt interface, we have chosen to configure the program such that users may easily make plain-language enquiries, for example, "Provide the anticipated wait time for 60 customers per hour." With the integration of this QA system, we aim to develop a comprehensive program that will assist the TSA throughout their entire operation. Specifically, we wish to provide TSA agents with waiting time estimates without requiring them to educate PaLM 2 or themselves on operations management intricacies. Furthermore, our envisioned chatbot version will conceal the technical complexities underpinning the model, rendering it significantly more user-friendly and intuitive.

This fusion of the two use cases seamlessly complements our existing Digital Twin project, enhancing its practicality. In essence, this entire project aims to diminish the necessity for specialized skills or domain knowledge from users, who are likely to be TSA agents. The ultimate objective is to employ plain English to construct an entire Digital Twin Project from the ground up that is not only functional but also user-friendly.

To achieve this, we entered a similar prompt into the Chat Prompt interface within MakerSuite.

Testing and Validation

Application 1: Codifying an Excel-based Operations Management model into Python3 through MakerSuite Implementation

Below are the steps we took to output a Python version of our Digital Twin Project through MakerSuite.

Step 1: **Convert our Excel Workbook into a text prompt**
After combining all the outputs and our Digital Twin models in Excel, we are able to produce a working prompt in English.

Step 2: Run the prompt into MakerSuite's text prompt interface

Step 3: Test the Python code generated by PaLM 2

Step 4: Cross-check the waiting time outputs with our Excel Models

Challenges we faced in harnessing the PaLM Suite

We focus on the challenges faced in deploying Application 1 (the task of “codifying” the Operations Management formulae), as they were significant and presented significant obstacles to Application 2 (developing a QA system based on that codified formula set.

Our challenge was in verifying that the PaLM Suite could produce code that would reliably produce the correct estimates, which we were unable to do. We elaborate on this in subsequent sections.

Challenges in codifying Operations Management formulae into Python3 code:

Throughout the project phase, we conducted more than 50 iterations of the final prompt before successfully obtaining the correct Python program output. Approximately 30% of these attempts resulted in PaLM misinterpreting the appropriate formulas, leading to negative waiting time estimations. To address this issue, we introduced a general formula as a precursor to detailing the specific formula for each stage of the process. For instance, we initially defined a general utilization formula before illustrating how to apply it to the ID Verification stage.

Out of these iterations, about 60% of the trials produced Python programs using the correct formulas but yielding inaccurate numerical results. Within this category of errors, nearly half stemmed from PaLM's difficulty in establishing the connection between different sections of the solution. To overcome this, we found that providing the full variable names before presenting the mathematical calculations significantly improved accuracy. The following two utilization examples serve to illustrate this point:

Utilization Factor for ID Verification Stage:
(Utilization for ID Verification Stage / (1 - Utilization for ID Verification Stage))
= 18.750% / (1 - 18.750%) = 23.08%

Example 1 (Redundant Version)

Utilization Factor for ID Verification Stage = 18.750% / (1 - 18.750%) = 23.08%

Example 2 (Concise Version)

The more verbose approach in Example 1 proved to be more effective than the concise version in Example 2. Therefore, it was necessary to be more explicit and elaborate on variable names for PaLM. The other half of this error category resulted from PaLM's inability to convert between seconds and hours, requiring the manual introduction of a new variable 'A' for this purpose.

Lastly, approximately 10% of the trials yielded Python code that failed to run at all. The output from this group primarily consisted of syntax errors in Python.

While it is undeniable that we could manually write the Python program in a fraction of the time it took to educate the LLM, we consider this experiment a significant advancement in the realm of Natural Language Processing (NLP). Employing an LLM to grasp intricate operational concepts and construct an operational model in Python is a task that individuals lacking knowledge of Python and Operations would likely find challenging. Furthermore, by harnessing PaLM, we have enhanced the reproducibility of our project, streamlining waiting time calculations compared to our Excel-based model. Consequently, this advancement has significantly increased the efficiency of our existing project. It is also noteworthy that we achieved the feat of performing mathematical computations without relying on an external calculator.

Implications for the discourse of LLM applicability in highly domain-specific applications

Some discourse has been directed towards the ability of LLMs trained on generalized corpora to predict the correct token outputs in highly domain-specific applications where the optimal probability distribution of the token-word set differs from that of general, lay-text owing to the importance of technical terms of art comprising the nomenclature of each specific domain.[1] We now add to the discourse by reporting the results of our attempts to assess the useability of the PaLM 2 Suite in the Operations Management domain.

1. *Ability to process arithmetic operations*

In the specialized domain of computational mathematics, the performance of even the most advanced large language models (LLMs) can falter during what may seem like elementary calculations. Our experiences with PaLM API and MakerSuite frequently

[1] Jaromir Savelka et al., “Can GPT-4 Support Analysis of Textual Data in Tasks Requiring Highly Specialized Domain Expertise?,” in *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 2023, 117–23, https://doi.org/10.1145/3587102.3588792.

highlighted this issue, revealing a distinct gap in basic arithmetic capabilities. To mitigate these shortcomings, we adhered to the suggested practice of embedding calculator-like notations within our prompts. Nevertheless, the results were often inconsistent, particularly when the LLM was tasked with executing multiple operations within a single formula.

Our workaround entailed breaking down complex equations into smaller, more digestible steps. As a toy example, the computation of $\frac{(a^2+b^2)}{2}$

Would be segmented into separate components: calculating a^2, then b^2, subsequently adding these values to arrive at $(a^2 + b^2)$, and finally dividing by 2 for the result.

2. *Prompt size restrictions*

Moreover, we grappled with the stringent character count limit during training—capped at 8196 for text prompts and 4096 for chatbot—that severely restrained the model's learning potential, especially for intricate tasks. This restriction led to a truncated learning experience, as the LLM could not be exposed to a sufficient variety of examples, thus impairing its computational understanding and output. To address this and make effective use of the limited character allowance, we condensed our prompts, attempting to supply the model with a denser array of examples. This condensation, however, forced us to omit certain contextual details that would have otherwise enhanced the model's performance.

3. *Temperature settings and reversion to the model's baselines for token prediction*

Adding to these challenges, we encountered issues with the model's pre-existing knowledge base, which would sometimes override the unique processes we were attempting to implement. The LLM's tendency to default to familiar algorithms, such as Little's Law, even when inappropriate for our specific context, further complicated training. We surmise that this is the result of the use of a generally trained corpus which would result in greater weight placed than desired on adjacent concepts that are not entirely relevant to the specific situation at hand. This, coupled with the character count limitations, posed substantial barriers to the nuanced training we envisioned.

4. *Overall comments*

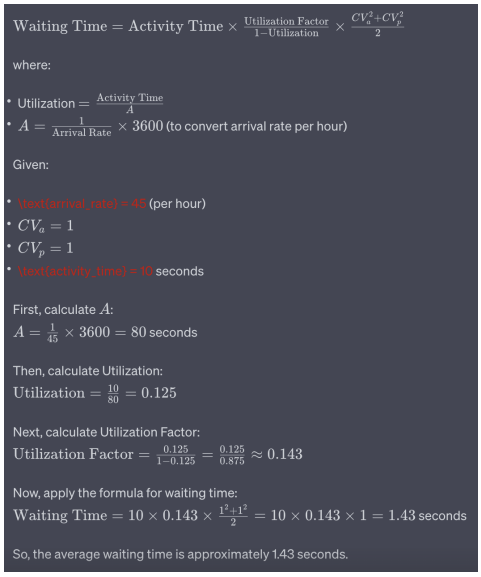
These collective hurdles accentuate significant constraints within the current human-computer interaction (HCI) framework. At the forefront of these is the LLM's inadequate execution of arithmetic operations, which forces users to craft prompts with exceptional precision and insight into the model's constraints to secure precise

outcomes. The character count limit only tightens these constraints, leading to a dilution of context and, consequently, a possible degradation in the granularity of data translation. These limitations demand an increased cognitive effort from the users, as they must engage in a nuanced process of prompt engineering to tailor their queries and instructions to fit within the model's constraints. The requirement for technical finesse from the users to effectively interact with the LLM is indicative of flaws within the HCI framework. The ultimate goal would be to minimize the gap between human interaction and machine interpretation to foster more intuitive and less burdensome interactions.

Implications for our Chatbot Implementation

Unexpectedly, while the Python script executed flawlessly on platforms such as ChatGPT, it did not perform as well when run on PaLM—specifically, the issue was isolated to the calculation outputs. Despite rigorous efforts to guide PaLM toward accurate computational results through various rewording strategies, these attempts did not yield the desired accuracy. Consequently, the decision was made to pivot and utilize another platform, leveraging the capabilities of GPT 4, to ensure the model's accurate performance.

To validate our code and confirm that there were no inherent errors, we conducted a test using both PaLM 2 and GPT 4. We posed a simple query to both systems: "What is the average waiting time given arrival_rate = 45, CVa = 1, CVp = 1, and activity_time = 10?" The results of this inquiry can be seen in the accompanying figures. The figure on the left depicts the correct output from ChatGPT, and the one on the right shows the incorrect result from PaLM 2.



$$\text{Waiting Time} = \text{Activity Time} \times \frac{\text{Utilization Factor}}{1-\text{Utilization}} \times \frac{CV_a^2 + CV_p^2}{2}$$

where:

- $\text{Utilization} = \frac{\text{Activity Time}}{A}$
- $A = \frac{1}{\text{Arrival Rate}} \times 3600$ (to convert arrival rate per hour)

Given:

- $\text{arrival\_rate} = 45$ (per hour)
- $CV_a = 1$
- $CV_p = 1$
- $\text{activity\_time} = 10$ seconds

First, calculate A:
$A = \frac{1}{45} \times 3600 = 80$ seconds

Then, calculate Utilization:
$\text{Utilization} = \frac{10}{80} = 0.125$

Next, calculate Utilization Factor:
$\text{Utilization Factor} = \frac{0.125}{1-0.125} = \frac{0.125}{0.875} \approx 0.143$

Now, apply the formula for waiting time:
$\text{Waiting Time} = 10 \times 0.143 \times \frac{1^2+1^2}{2} = 10 \times 0.143 \times 1 = 1.43$ seconds

So, the average waiting time is approximately 1.43 seconds.

```
1. **Calculate A**

   ```
   A = (1 / 45) * 3600 = 80
   ```

2. **Calculate utilization**

   ```
   utilization = 10 / 80 = 0.125
   ```

3. **Calculate utilization factor**

   ```
   utilization_factor = 0.125 / (1 - 0.125) = 0.15625
   ```

4. **Calculate waiting time**

   ```
   waiting_time = 10 * 0.15625 * ((1**2 + 1**2) / 2) = 1.953125
   ```

5. **The final answer is 1.953125.**
```

As demonstrated, the results from GPT 4 are in line with our expectations, while those from PaLM 2 indicate discrepancies. This discrepancy led us to conclude that the model operates more effectively on the GPT 4 platform, prompting our transition to this service for further computations.

Phase 4: Streamlining User Interactions via Chatbot Integration

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. 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.

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.

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:

If the bottleneck is resolved and all available staff have been allocated, the system outputs the updated waiting time with these new allocations.
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.
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. 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. 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.

GPT 4

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. 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. 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.

Learnings & 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.

VI. Insights & Applications of Model

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.

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.

VII. Future applications

The TSA Security screening model, known for its multi-level checks and buffers, holds potential for adaptation in various settings beyond airport security, especially where security and efficient flow are essential. This model's success in enhancing safety and customer satisfaction is evident in settings like concerts and theme parks. However, its application to more complex environments, such as healthcare, requires nuanced and detailed adaptations due to the higher stakes involved.

Similar to TSA checkpoints, healthcare visits involve multiple steps: check-in at reception, doctor consultation, and receiving treatment or a prescription, each

punctuated by significant wait times. However, the unique nature of each patient's condition, requiring individualized care paths, adds considerable complexity compared to the more uniform process at TSA checkpoints. The gravity of potential errors in healthcare, directly impacting patient health and safety, demands a more careful and specialized approach in model application.

The TSA model's principle of managing flows and minimizing wait times can be extended to other linear process settings. For example, fast-food restaurants, where the flow is akin to an assembly line, can benefit from this model to enhance efficiency. Theme parks, where ride durations are consistent but variations occur due to downtimes and breakdowns, also present a suitable application scenario. In these contexts, parameters like activity times and multi-server queuing systems can be directly applied, simplifying the adaptation process.

To further enhance the applicability of our model, incorporating QA systems would be a significant improvement. In settings like concerts and theme parks, QA systems can provide real-time wait time updates and identify bottlenecks, suggesting staff reallocation to reduce congestion. In healthcare, such systems could assist in managing patient flow, advising on staff distribution based on current wait times. This integration not only optimizes operations but also significantly improves customer and patient experiences, demonstrating the adaptability and effectiveness of our model across a range of environments.

VIII. Appendix

This appendix has been redacted to protect the confidentiality of the project's content.

IX. 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/

Bullock, D., Haseman, R., Wasson, J., & Spitler, R. (2010). Automated Measurement of Wait Times at Airport Security: Deployment at Indianapolis International Airport, Indiana. Transportation Research Record: Journal of the Transportation Research Board, 2177(1). https://journals.sagepub.com/doi/abs/10.3141/2177-08

Cabanatuan, M. (2022, June 15). SFO says passengers should prepare for hours-long waits at security amid surge in travel. San Francisco Chronicle. https://www.sfchronicle.com/bayarea/article/SFO-says-passengers-should-prepare-for-hours-long-17242313.php

Cachon, G. (n.d.). Matching Supply with Demand. In Matching Supply with Demand.

Elliott, C. (2018, March 25). Travel nightmares: What to do if you miss your flight. USA Today. https://www.usatoday.com/story/travel/advice/2018/03/25/missed-flight-passenger-rights/450366002/

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

Frey, M., Kiermaier, F., & Kolisch, R. (2017). Optimizing Inbound Baggage Handling at Airports. Transportation Science, 51(4). PubsOnline. https://pubsonline.informs.org/doi/abs/10.1287/trsc.2016.0702

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

Gritzka, K., Niemeier, D., & Mannering, F. (2006). Airport security screening and changing passenger satisfaction: An exploratory assessment. Journal of Air Transport Management, 12(5), 213-219. https://www.sciencedirect.com/science/article/pii/S0969699706000354

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the Digital Twin: A systematic literature review. Elsevier CIRP Journal of Manufacturing Science and Technology, 29(A), 36-52. https://doi.org/10.1016/j.cirpj.2020.02.002

Leone, K., & Liu, R. (. (2011). Improving airport security screening checkpoint operations in the US via paced system design. Journal of Air Transport Management, 17(2), 62-67. https://doi.org/10.1016/j.jairtraman.2010.05.002

Naji, M., Braytee, A., Al-Ani, A., Anaissi, A., Goyal, M., & Kennedy, P. J. (2020). Design of airport security screening using queueing theory augmented with particle swarm optimisation. Service Oriented Computing and Applications volume, 14, 119-133. https://link.springer.com/article/10.1007/s11761-020-00291-0

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). 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

Sakano, R., Obeng, K., & Fuller, K. (2016). Airport security and screening satisfaction: A case study of U.S. Journal of Air Transport Management, 55(August 2016), 129-138. https://www.sciencedirect.com/science/article/pii/S0969699716300564

Shaw, K. V. (2016). Airports Leverage Technology to Take the Mystery Out of Security Wait Times. Airport Improvement Magazine. https://airportimprovement.com/article/airports-leverage-technology-take-mystery-out-security-wait-times

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

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

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

Wang, H., Liu, K., Qi, Z., & Guo, J. (2017). The establishment and optimization of airport security check process model. 2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC). IEEE. https://doi.org/10.1109/YAC.2017.7967561

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.ncbi.nlm.nih.gov/pmc/articles/PMC3123545/

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.

David, R., Park, S., Kang, H., Ho, S., Kim, J. (2023). A Prototypical Full-Stack Digital Twin Implementation for Optimizing Airport Security Screening Processes.

Disclaimer and Legal Notice

The information and analysis presented in this document are provided solely for academic and research purposes. It is imperative to recognize that this work does not constitute an official evaluation, endorsement, or representation of real-world operations, policies, or practices of any organizations, government agencies, or entities mentioned herein.

The authors of this paper have strived to ensure the accuracy and reliability of the content to the best of their abilities, but they do not assume liability for any errors, omissions, or interpretations that may arise from the utilization of this material.

Furthermore, any resemblances to actual events, scenarios, or individuals, whether living or deceased, are purely coincidental and unintentional.

Readers are encouraged to independently verify and validate the information contained within this document, and they should not rely solely on its contents for any decision-making processes. This paper should be perceived as a contribution to academic discourse and intellectual exploration rather than an authoritative source of information.

Lastly, it is essential to acknowledge that the opinions, viewpoints, and conclusions expressed herein solely belong to the authors and do not necessarily reflect the policies, positions, or endorsements of any academic institutions or organizations associated with the authors.

By accessing and using this document, you explicitly acknowledge and accept the terms of this disclaimer.