Digital Object Identifier

Group 15 Report

OGHENEMALU F. IGHOIYE, MOHAMMED A. CHOUDHARY, YIFAN WU, COLTON T. GEBBEN, QINGHAN HUANG AND ZHAOHUAN GUO.
Group 15

ABSTRACT The rise of overtourism threatens the attraction of tourist destinations, especially in urban areas like Manhattan. Our application, "City Pulse Manhattan", tackles this by offering tourists real-time predictions of busyness levels at city attractions. Built on a decoupled architecture, the application promotes efficient data interaction through RESTful APIs. Back end operations, boosted by Spring Boot, ensure performance and scalability. Meanwhile the front-end, utilising Vue.js, Element Plus, and ECharts, provides an optimized UI and vivid visualisations. Enhancements like responsive design, lazy loading, and on-demand imports further refine user experience. In our data analysis, Yellow Taxi Trip datasets, NYC Taxi Zones, and historical weather data were utilised. Through methodical data analysis, we identified spatio-temporal trends in taxi usage, and we used this as a metric for “busyness”. Multiple modeling runs were carried out, with the Random Forest model from the 2019 dataset showcasing the best results. Evaluations confirm our model’s aptitude to reliably predict tourist attraction busyness, offering a novel solution to manage overtourism.

Website: Visit our project website at http://csstudent06.ucd.ie/.
GitHub Repository: https://github.com/QinghanHuang/NewYork-Busyness-Web

I. INTRODUCTION

Tourism is pivotal to the global economy, accounting for roughly 9.8% of the global GDP and 7% of worldwide exports [1]. It notably augments local economies by enhancing foreign exchange reserves through market competition [2] and ushering in new infrastructural investments [3]. The industry also drives commercial expansion [4], generates job opportunities, and elevates local income levels [4]. As economies prosper, the subsequent growth of the tourism sector can catalyze further increases in international travel [5].

Nevertheless, the adverse effects of tourism, termed overtourism, become prominent when tourist influx surpasses a location’s capacity [6]. This phenomenon compromises the residents’ quality of life, negatively influences tourist experiences [7], dampens tourist loyalty [8], and can adversely impact the local economy [9], overall tourist satisfaction [10], and intentions to revisit [11]. It also poses environmental hazards [12].

New York City, a global epicenter of finance and culture, confronts the repercussions of overtourism, especially in Manhattan. As the most populous and frequently visited city in the USA, boasting approximately 8.8 million inhabitants and 10 million annual tourists, its vibrant ambiance can sometimes hinder optimal experiences for both residents and visitors. An application that forecasts the crowd levels at tourist spots could augment the allure and accessibility of these locales, enriching the city’s overall visitor experience.

Our proposed app, City Pulse Manhattan, is designed to offer real-time congestion predictions, termed busyness, for diverse Points of Interest (POIs) across the city. Considering the city’s bustling nature, about 56% of its workforce is dependent on public transit for daily commuting [13]. Hence, tapping into public transport data, especially Yellow Taxi records, can shed light on real-time location-based activity hubs. Initially, the app will focus on Manhattan, a major taxi hotspot. While taxis aren’t the primary transit method and serve as an auxiliary mode, their adaptability makes them distinct, given they aren’t bound by the same infrastructural constraints as buses or trains. Still, their usage is substantial, with 2019 alone seeing an average of over a million daily taxi rides [14].

The application’s objective is to help travellers, locals, and policymakers by forecasting the level of activity at Manhattan POIs in order to alleviate overtourism issues. This strategy will promote sustainable urban development, create a balance between the quality of life of locals and tourism expansion, and improve traveller experiences.

II. LITERATURE REVIEW

Human movement pattern is often a complex phenomenon. However, several studies have demonstrated human mobility exhibits significant spatio-temporal reoccurrence [15, 16]. This confirms human mobility trends can often be predicted using historical data [17], essentially providing vital information to researchers, particularly in understanding population distribution and overcrowding [18], urban development [19], and geo-spatial services [17]. Several studies have tried to accurately predict tourism demand or taxi demand across several geographical locations. While these studies are not directly measuring the busyness, their results could be considered analogous to busyness, as they reflect the predicted level of human activities in their study locations. These studies typically employ a wide range of models such as timeseries, linear regression, and artificial neural networks to forecast demand or flow.

A. TOURISM DEMAND

The Autoregressive Integrated Moving Average (ARIMA), in particular, is a dominant timeseries forecasting model, widely used in several domains, including tourism with good results [20, 21]. These models are often utilised and justified in tourism research as they address the phenomenon of seasonal variations [22]. Recently, it is becoming more prevalent in modern tourism research to incorporate tourist search results or online behaviour as an explanatory variable in these models [23]. For instance, Xiankai Huang et al [24] conducted a study that showed augmenting ARIMA models with search results led to a significant increase in accuracy, with an improvement of 12.4%. Several other models have been used to predict tourist demand, with more intelligent models like Artificial Neural Networks (ANN) being preferred. While some researchers have shown ARIMA models outperform ANNs [25], results from others studies suggest otherwise [26, 27], with ANNs outperforming Express Smoothing (ES), ARIMA, and multiple regression [27]. One study also tried to predict the probability of overcrowding on a beach by analysing photos and smartphones [28].

B. TAXI DEMAND

Attempts at predicting taxi demand across various locations has also utilised different models such as ARIMA, ANN, and Long Short Term Memory (LSTM). LSTMs in particular have been shown to outperform ARIMA models. Taxi demand is not only affected by spatial factors like pickup or drop-off location, but also temporal factors like pickup or drop-off time, weather, and special events [29]. In their study Jiang and Zhang [30] utilised geospatial data, represented as an image, and a residual network to predict taxi pickup and drop-off demands in New York City.

Considering the impact of the Covid-19 pandemic is also important in understanding changes in taxi demand. For instance, Sen Li et al [31] compared spatial and temporal differences in demand before and after the pandemic. When compared to pre-pandemic levels in New York, they estimated taxi journeys reduced by 95.3% one week post restrictions. However, they noted the spatial and temporal clustering of these taxis changed as well [31].

III. METHODOLOGY

The web application employs a decoupled architecture that facilitates efficient data exchange between the front-end and back-end components through RESTful API interfaces in JSON format. This design approach enables parallel development by front-end and back-end teams. It fosters a focused approach to their responsibilities, thereby significantly enhancing overall development efficiency and making maintenance easier. This section will introduce the frameworks, technologies, and tools utilized in front-end and back-end development.

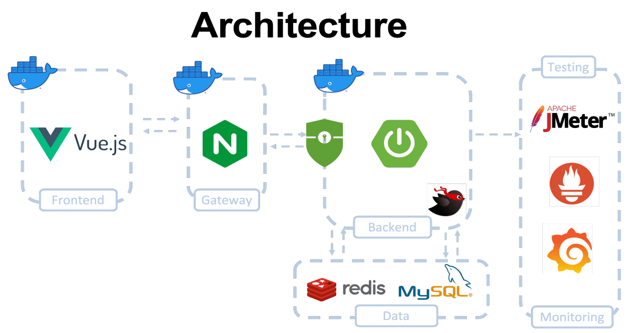


FIGURE 1. Project Architecture Diagram

A. THE BACKEND

1) Backend Framework

The backend leverages the Spring Boot framework, adopting the MVC architecture. We chose Spring Boot for three primary reasons. First, its exceptional performance capabilities empower the system to manage high levels of concurrent operations and respond swiftly to user requests. Second, the framework's extensible modules, which encompass web frameworks, security components, and Prometheus health monitoring, significantly bolstered the robustness of our application. Third, the ease of deployment and inherent horizontal scalability of Spring Boot provided a seamless path for future expansion in response to emerging demands.

2) Cache Mechanism & Asynchronous Tasks

During the development process, after obtaining the initial MVP, we encountered the challenge of slow response times. This was attributed to the need for external APIs, such as weather data, and the requirement to load models for predictions. We took measures to address this problem.

Inspired by the caching concepts in computer networks, we sought to incorporate a caching mechanism into our project. Following thorough study and research, we successfully integrated a caching mechanism into our project, resulting in enhanced performance.

The SQL database, established through AWS RDS, is a MySQL database utilised for storing persistent data, such as encrypted user information and Places of Interest details. Besides, the MyBatis framework is employed to map the SQL query results to corresponding objects.

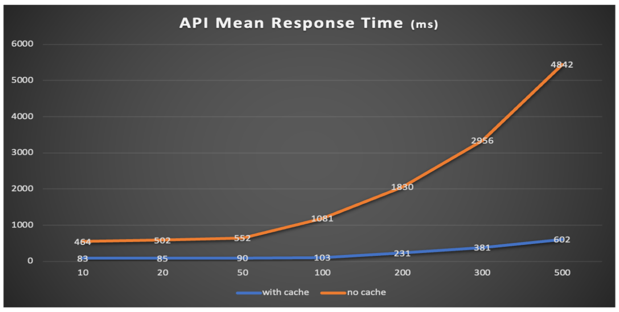


FIGURE 2. Comparison Chart of Response Times with and without Cache Mechanism

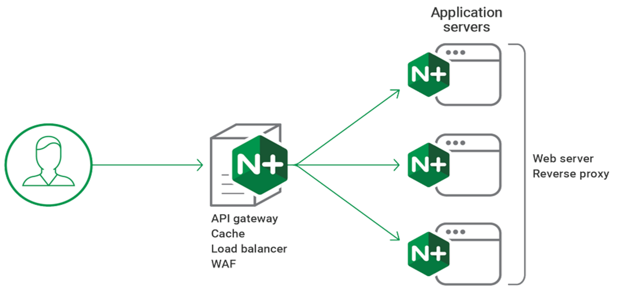


FIGURE 3. Nginx Reverse Proxy Illustration (from the official website)

For storing temporary data and introducing caching, we also incorporated Redis, a NoSQL database. Within our Spring Boot application, we established two scheduled tasks to fetch weather data and run the prediction model. Every hour, a six-day weather forecast is fetched from the Open Weather API, and the result is uploaded to Redis. Every six hours, the congestion levels for the coming 3 days predicted by the model are uploaded to Redis as well. This approach ensures data is up-to-date and leverages the caching mechanisms to optimize response speed when multiple users are accessing our service.

3) Network Security

In the final section of 'Computer Networking', a module at University College Dublin, Professor McArdle delved into 'Network Security', providing us with a comprehensive overview of the significance of network security and potential types of network attacks, greatly enhancing our understanding of network security. Safeguarding the security of our servers is a crucial responsibility and we gained insights into the most likely attacks a server may face and the corresponding solutions. We applied this knowledge to our project.

A DDoS attack occurs when a target server or service is flooded with a massive volume of requests at once, rendering it inaccessible to legitimate users. In order to mitigate this threat, we utilise Nginx's reverse proxy functionality, rendering the backend server invisible to external entities. This reduces the likelihood of attackers bypassing the frontend and directly targeting the server, while also safeguarding the server-side code from being easily compromised.

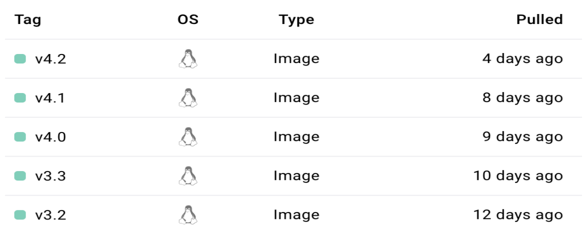
Another potential network threat we anticipate is Cross-Site Request Forgery(CSRF) attacks and cross-origin access. We applied Spring Security's safeguard mechanisms against CSRF attacks. This implementation adds an extra layer of security, requiring authentication for data retrieval whether accessed through the frontend or via direct server queries. This significantly enhances the server's security, effectively acting as a lock that only authorized entities can access data. Furthermore, we diligently enforce strict cross-origin restrictions on the backend server, allowing only the IP address of the frontend server to access the backend.

By implementing the aforementioned methods, we added three layers of 'security lock' to our server. Since network security is a relative concept, there is unfortunately no absolute assurance. Consistent practice is needed to uncover system vulnerabilities and ensure the server remains at a higher level of security.

4) CI/CD & Scalability

Following the MVP principle, it is crucial to have a runnable and deployable project at each stage. This implies the practice of continuous development and deployment, aligning with the principles of Continuous Integration and Continuous Deployment (CI/CD).

For our project, containerized deployment is the ideal solution, and we utilised Docker for containerisation. Throughout our development process, after achieving each stage's MVP, we utilise Dockerfiles to build our Dockerimages. These images are then continuously deployed by pulling them directly from Docker Hub onto the server. Additionally, we employ Docker Compose for lightweight container orchestration.



Tag	OS	Type	Pulled
● v4.2	🐧	Image	4 days ago
● v4.1	🐧	Image	8 days ago
● v4.0	🐧	Image	9 days ago
● v3.3	🐧	Image	10 days ago
● v3.2	🐧	Image	12 days ago

FIGURE 4. Part of 26 Historical New York Business Project Docker Images

In developing our current project, it has been essential to consider potential future growth in business and traffic. Therefore, scalability is a crucial consideration. Horizontal scaling emerges as the predominant solution, striking a delicate balance between cost-effectiveness and performance.

After researching various mature solutions, our approach involves integrating Spring Cloud, Docker, and Nginx. We took these factors into account when initially designing the project architecture. We use Docker's horizontal scaling capabilities to establish a distributed server cluster. Additionally we transformed the project into a microservices architecture using the Spring Cloud framework and use Nginx for load balancing.

While we have not applied the aforementioned three components directly to the current project, we have gained extensive theoretical knowledge of each module and carried out practical demonstrations through demos. These serve as practical solutions for future expansion and we took them into account during initial considerations. With these solutions in place, a straightforward horizontal scaling can significantly enhance our business's throughput in the future, without major architectural overhauls.

5) API Documentation

For interface development, we employed Swagger to generate API documentation and Postman is utilized for API testing. Swagger enhances communication and collaboration between frontend and backend teams in a decoupled project.

B. THE FRONTEND

1) Frontend Framework

Our frontend was primarily built on the Vue.js framework, chosen for its progressive nature, Virtual DOM, and ease of gradual adoption. By integrating Axios, we facilitated robust HTTP requests, ensuring a distinct separation between front-end and back-end. Our adherence to RESTful API principles enhanced data exchange, ensuring a clean, maintainable codebase.

2) Responsive Design

User adaptability was at the core of our design philosophy. From the outset, we planned for dual user interfaces catering to different screen sizes, enhancing the browsing experience. Features like the 'POI detail' were optimised for mobile views, sliding up from the bottom, exemplifying our dedication to user-centric design across devices.

3) UI Library

Element Plus, with its extensive components and compatibility with Vue.js, was our primary UI library. For data representation, ECharts was adopted due to its interactive capabilities, allowing us to vividly depict tourist attraction congestion levels. Both choices elevated the user experience, making data interpretation intuitive.

4) Performance Enhancement

Optimising user experience was paramount. Implementations like lazy loading were incorporated to load resources only when essential, ensuring swift page rendering. "On-demand imports" were another highlight, where components were dynamically loaded to minimise initial load times. Additionally, Vue's reactivity system and lifecycle hooks were harnessed to regulate re-renders, maximising system efficiency.

IV. DATA ANALYTICS & VISUALISATION

The Data Analytics segment of this project involved data sourcing, cleaning, processing, feature engineering, and modeling.

A. DATA SOURCING

We utilise the following datasets: the 2019 and 2018 Yellow Taxi Trip Data [32][33], NYC Taxi Zones dataset [34], and the Openweather historical weather dataset [35]. These datasets provide intricate details about NYC taxi trips, including pickup and drop-off times, locations, payment details, and more. For a comprehensive list of features and definitions, refer to the data dictionaries [32][33]. We disqualified the 2020 and 2021 Yellow Taxi Trip datasets due to possible distortions on taxi usage by the pandemic.

We used the taxi zones dataset to filter out data relevant to Manhattan. It maps location IDs from the Yellow Taxi datasets to their corresponding boroughs in NYC. This dataset also contains dimensions of the taxi zones, which we incorporate into the model. The Openweather dataset offers historical weather data of New York City, including many weather features such as temperature, precipitation, and wind-speed [35].

B. DATA CLEANING & PROCESSING

We used Python's Pandas library [36] as our main tool for data cleaning and processing. For a comprehensive set of methods used in the data cleaning and processing, refer to our Github repository [37].

We generate data quality reports to identify missing data, outliers, negative values, and incoherent values such as distances of 0. Table 1 below provides an overview of part of the data quality report [37].

Feature	Outliers	Missing Values	Negative Values
VendorID	n/a	163435	0
tpep_pickup_datetime	n/a	0	n/a
tpep_dropoff_datetime	n/a	0	n/a
passenger_count	6822193	163435	0
trip_distance	8616584	0	6379
RatecodeID	n/a	163435	0
store_and_fwd_flag	n/a	163435	n/a
PULocationID	n/a	0	0
DOLocationID	n/a	0	0
payment_type	n/a	163435	0
fare_amount	7245537	0	138930
extra	115547	0	69385
mta_tax	526965	0	136433
tip_amount	3377959	0	763
tolls_amount	4674154	0	3101
improvement_surcharge	166209	0	138742
total_amount	7254872	0	138908
congestion_surcharge	599687	4585780	118992

TABLE 1. Data Analysis Summary[37]

C. FEATURE ENGINEERING

We used data visualization tools, like bar charts for categorical data and histograms for continuous data [37]. We derived features, including the day of the week, month, day of the month, and hour, from date-time to highlight temporal trends in the target busyness feature, since many machine learning algorithms are not able to process date-time features directly [38]. We created other new features, such as journey duration and average speed, and note their significance in the feature importance document [37]. During our feature engineering process, we encountered an issue with the discrepancy in row counts between the data rows and the rows for the target we created. To resolve this we aggregated the rows to one row per hour. We took the mode for categorical features and the mean or median for continuous features in the aggregate rows. Categorical features underwent one-hot encoding [39], and continuous features were scaled for some training runs depending on which algorithm we used. We gauged busyness using taxi trip density and computed it as:

$$\text{busyness} = \frac{\text{number of taxi trips}}{\text{time} \times \text{area of zone}} \tag{1}$$

We favored hourly predictions to speed up training time and since we did not expect busyness levels to change at higher temporal resolutions based on the hourly busyness variation [37]. The area sizes came from the taxi zones dataset [34] and we concatenated them into the main dataset. The target was split into five categories, labeled 1-5, where higher numbers denote greater busyness. Each label covers 20% of the data rows.

Run	Purpose
1	Random Forest
2	Random Forest + Weather
3	Random Forest (2018+2019)
4	Random Forest (Feature changes)
5	Random Forest (Hyperparameters)
6	Random Forest (deployment model)
7	Logistic Regression
8	K Nearest neighbour
9	Decision tree classifier
10	XGB classifier
11	Neural network
12	Random Forest (deployment model with weather)

TABLE 2. Training Runs

D. MODEL TRAINING & EVALUATION METRICS

Twelve training runs were undertaken. We performed the initial six runs (1 and 7-11) for algorithm selection. We trained many different algorithms, such as Random Forest, Logistic Regression, K- Nearest Neighbour, Decision Tree Classifier, XGB Classifier and Neural Network, in order to contrast the algorithms and choose the most suitable one for our purposes. We split data 80% for training and 20% for testing, with the test data representing a consecutive time series instead of a random sample. This format showcases the model's ability to predict over extended periods and demonstrates an ability to forecast sequentially. Evaluation benchmarks chosen for multi-class classification include accuracy, recall, precision, and the F1 score [40].

1) **Accuracy**:
   - Definition: Proportion of all predictions that are correct[40].
   - Formula[40]:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} \tag{2}$$

2) **Precision**:
   - Definition: Proportion of positive identifications that were correctly classified[40].
   - Formula[40]:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3}$$

3) **Recall**:
   - Definition: Proportion of actual positives that were identified correctly[40].
   - Formula[40]:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4}$$

4) **F1 Score**:
   - **Definition:** The F1 score is a measure of accuracy that considers a harmonized balance of recall and precision [40].
   - **Formula[40]:**

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5}$$

Run	Accuracy	Precision	F1	Recall
1	0.8523	0.8522	0.8519	0.8521
2	0.8319	0.8313	0.8308	0.8317
3	0.8491	0.8491	0.8499	0.8524
4	0.8547	0.8546	0.8544	0.8545
5	0.8537	0.8536	0.8534	0.8536
6	0.7391	0.7380	0.7337	0.7378
7	0.7188	0.7178	0.7080	0.7179
8	0.8283	0.8302	0.8282	0.8287
9	0.7879	0.7869	0.7871	0.7877
10	0.8322	0.8314	0.8312	0.8318
11	0.8850	0.8875	0.8855	0.8851
12	0.7074	0.7097	0.6984	0.7058

TABLE 3. Training Runs with All Metrics

Random Forest and neural network models topped the accuracy charts. We selected the former for its compatibility with the backends required PMML file format. Subsequent runs explored the influence of weather data, additional years of data, more decision trees, and feature importance-based selection [37]. However, we observed no significant improvements. The final model solely relied on the 2019 Yellow Taxi Trip dataset [32]. The 6th and 12th training runs, tailored to meet backend file size constraints, employed a smaller Random Forest Classifier with fewer features as live data is not currently available for all features in the dataset, making such features unusable for live and future predictions. Consequently, their accuracy suffered and in future work we hope to employ more accurate versions of the model with fewer hardware constraints. The 2nd and 12th runs revealed weather data's detrimental effect on prediction quality while the 3rd run demonstrated that extra taxi data did not enhance accuracy. Finally, runs 4 and 5 indicated that prioritizing features and increasing decision trees had negligible effects on accuracy. In summary, we successfully explored, investigated, and cleaned the datasets. Our models had high performance on accuracy and other benchmarks and integrated seamlessly with our application.

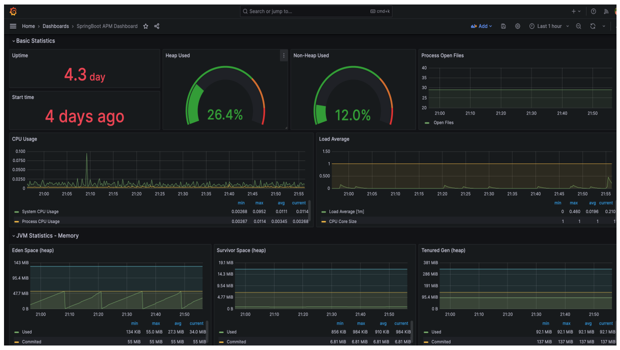
V. EVALUATION & RESULTS

Our project evaluation emphasizes three key areas: server system metrics, stress testing, and user feedback. By analyzing these components, we intend to understand our system's resilience, performance, and user satisfaction, informing our project development decisions.

A. SERVER SYSTEM METRIC VISUALIZATION

After deployment, it is crucial to implement server metric monitoring. While traditional logging systems such as Log4j are effective, they miss an essential visual element that is intuitive and easily shared.

In light of this, we employed the synergy of Prometheus and Grafana. Prometheus, adept in data collection and querying, captured various metrics from our system. Grafana, renowned for visualization, constructed interactive dashboards for these metrics.



CPU Usage	min	max	avg	current
System CPU Usage	0.00268	0.0952	0.0111	0.0114
Process CPU Usage	0.00267	0.0114	0.00345	0.00268

Load Average	min	max	avg	current
Load Average [1m]	0	0.460	0.0196	0.210
CPU Core Size	1	1	1	1

Eden Space (heap)	min	max	avg	current
Used	134 KiB	55.0 MiB	27.3 MiB	34.0 MiB
Commited	55 MiB	55 MiB	55 MiB	55 MiB

Survivor Space (heap)	min	max	avg	current
Used	856 KiB	984 KiB	910 KiB	984 KiB
Commited	6.81 MiB	6.81 MiB	6.81 MiB	6.81 MiB

Tenured Gen (heap)	min	max	avg	current
Used	92.1 MiB	92.1 MiB	92.1 MiB	92.1 MiB
Commited	137 MiB	137 MiB	137 MiB	137 MiB

FIGURE 5. New York Business Project Visual Dashboard for System Metrics Obtained by Prometheus and Grafana

The adoption of Prometheus and Grafana brought forth a host of benefits for our project:

Visual Clarity: Complex server metrics transformed into intuitive visuals, granting insight into system health and performance.

Real-time Insights: The real-time monitoring offered by this solution enabled us to detect anomalies promptly and take proactive measures, ensuring seamless operation and swift issue resolution.

Collaborative Decision-Making: The visual nature of Grafana dashboards enabled easy sharing and collaboration among team members, promoting informed decision-making and cohesive problem-solving.

Important Indicators: As our project scaled, Prometheus and Grafana seamlessly accommodated increased data volumes and growing monitoring needs, offering crucial indicators for assessing the need for server expansion.

B. PERFORMANCE TESTING

After establishing system metric visualization, our focus shifted to stress testing to replicate spikes user visits to our dashboard. This process evaluated our system's robustness under varying user loads.

1) Stress Testing Using Visual Metrics

Using visual metrics from Prometheus and Grafana, we simulated high user activity scenarios. This not only tested the system's durability but also spotlighted potential weak points.

2) Utilizing Key Stress Testing Tools: JMeter

There are several tools available for stress testing with two prominent options suitable for the Spring platform, JMeter and JMH. JMeter primarily conducts performance tests on API responses, while JMH focuses on evaluating the execution time of different methods within Java programs.

After comparing JMeter and JMH for stress testing, we chose JMeter. Its focus on end-to-end performance testing made it the ideal choice for our project's needs. Its versatility allowed us to simulate various user interactions, generating realistic loads and providing accurate insights into our system's behavior under stress.

3) Analyzing Response Times & Calculating Concurrency

During stress testing, we meticulously analysed the interface's response times, measuring the system's ability to handle increasing loads. With a clear understanding of the testing tool's analytical methodology, we opted for multiple test runs using distinct configurations. Leveraging our efficient caching mechanism, the majority of interfaces exhibited closely matched and rapid response times. Below are the calculated response metrics across different TPS scenarios based on several rounds of testing with the final values representing the averages.

The results reveal that the performance of the server we employed remains commendable up to a TPS of 500, with response times showing minimal increase. However, beyond 500 TPS, there is a significant surge in response time and error rates, nearly reaching an unusable state. Consequently, our analysis and calculations indicate that the system can sustain a concurrency of 500 interface requests per second for a single endpoint.

TPS	Average	Mean	90 Line	Error
10	13	14	14	0.00
20	12	13	15	0.00
50	14	12	13	0.00
100	16	17	12	0.00
200	34	24	80	0.00
300	91	21	251	0.00
400	98	33	290	0.00
500	993	317	2070	0.08
600	3287	3045	4125	7.62

TABLE 4. Interfaces Metric for Different TPS

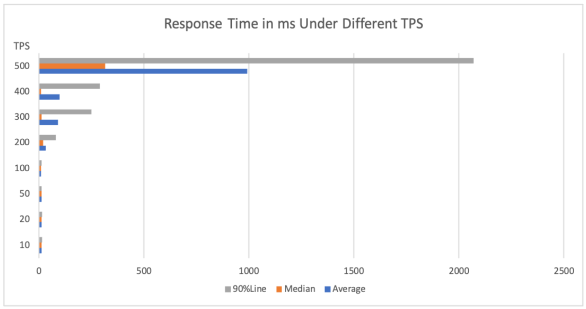


FIGURE 6. Response Time in ms for Different TPS

4) Deriving User Capacity Formula

Through stress testing and concurrent user calculations, we derived a valuable formula to estimate user capacity. This formula allowed us to determine the maximum number of users the system could handle per minute under varying loads, aiding capacity planning and scalability decisions.

The formula for calculating the number of users that can be sustained per minute is:

$$\text{Maximum Users/min} = \frac{\text{Max concurrent requests /s} \times 60}{\text{Average interface calls/min}} \tag{6}$$

Based on the analysis of the project code and network packet capture tools, we observed that during the initial load of the project, 8 interfaces need to be called. Subsequently, as users engage with the application, an average of 3 interface calls occurs every 2 seconds. Combining these observations, conservatively estimating, we can calculate that users would likely make approximately 100 interface calls per minute during their usage.

By combining the concurrency results we obtained from JMeter and the calculations we provided earlier, when plugging in the values into the formula, we calculate that this project can accommodate approximately 300 users per minute on a single-core Linux server with 2GB of memory. This metric holds significant importance as it serves as a crucial reference for determining whether horizontal scaling is needed and aids in troubleshooting system failures.

C. USER SURVEY

Our user evaluation, comprising both mobile and desktop users, highlighted the system's strengths and areas for enhancement. Unfortunately, our survey couldn't distinguish feedback based on user device types effectively, which could have indicated if certain features were received differently on different devices.

A strong 80% of participants commended the system's usability and responsiveness, and 70% found selecting a Point of Interest (POI) straightforward. While the interface was well-received by 80%, 10% suggested layout tweaks.

70% of users appreciated our data representation, confidently gauging Manhattan's busyness, while 20% desired clearer visual indicators for congested areas. 80% were confident in the data's timeliness.

The filter/search function was valued by 90%. However, opinions on route planning were split: 50% found it user-friendly, but 30% didn't. Similarly, while 60% considered the instructions clear, 30% thought otherwise.

When comparing our platform to others, 30% viewed ours as superior, but 20% disagreed. Nonetheless, 70% agreed our system fulfilled their expectations.

In summary, the feedback was largely positive, emphasizing enhancements in data visualization, guidance, and route planning. Future evaluations aim to provide more device-specific insights.

VI. FUTURE WORK & CONCLUSION

A. FUTURE WORK

Our study's findings illuminate paths for further refinement and investigation. One challenge we encountered is the latency during data transfer, attributed to the substantial boundary points for each taxi zone. Transitioning to the GeoJSON format, tailored for geographical features, could optimize this transfer and grant users a clearer comprehension of taxi zones.

Regarding the user interface, introducing a log-in free mode could expand accessibility. This mode would allow user interaction without compromising user data security or the system's integrity. Moreover, improving itinerary adjustments with a click-and-drag mechanism would simplify the reordering process.

Diversifying our prediction algorithm is another promising avenue. Incorporating licensed datasets, such as mobile phone data, could provide unparalleled insights into real-time movements, augmenting our current taxi-centric model.

In essence, while our framework is robust, the horizon holds several promising avenues. Embracing advanced data formats, refining user-driven features, and leveraging richer datasets can further enhance our platform's user experience and analytical prowess.

B. CONCLUSION

In the dynamic landscape of global tourism, Manhattan stands out both as a beacon and a challenge. In confronting the challenges of overtourism, our application, City Pulse Manhattan, emerges as an inventive solution.

By utilizing the Yellow Taxi dataset, we tapped into Manhattan's transit dynamics to predict congestion at city POIs. This data-driven approach yielded patterns that validated the link between taxi movement and tourist attraction busyness. Rigorous data analysis and algorithmic adjustments heightened the accuracy of our predictions.

Our evaluations spanned from understanding user perceptions to assessing system performance. Positive user feedback attested to City Pulse Manhattan's value, while server metrics and stress tests vouched for its robust technical capabilities.

Further elevating the user experience, POI information profiles and an itinerary tool equip users with beneficial navigation aids. These tools highlight the app's worth as both a navigation resource and a blueprint for urban strategising.

In conclusion, City Pulse Manhattan represents a notable stride towards balancing tourism demands with the well-being of urban residents. With continued development, it possesses the potential to set a benchmark for sustainable urban tourism worldwide.

REFERENCES

[1] C. F. Tang, A. K. Tiwari, and M. Shahbaz, "Dynamic inter-relationships among tourism, economic growth and energy consumption in India," Geosystem engineering, vol. 19, no. 4, pp. 158-169, 2016.
[2] R. I. McKinnon, "Foreign exchange constraints in economic development and efficient aid allocation," The Economic Journal, vol. 74, no. 294, pp. 388-409, 1964.
[3] A. Blake, M. T. Sinclair, and J. A. C. Soria, "Tourism productivity: evidence from the United Kingdom," Annals of Tourism Research, vol. 33, no. 4, pp. 1099-1120, 2006.
[4] C.-C. Lee and C.-P. Chang, "Tourism development and economic growth: A closer look at panels," Tourism management, vol. 29, no. 1, pp. 180-192, 2008.
[5] J. G. Brida, I. Cortes-Jimenez, and M. Pulina, "Has the tourism-led growth hypothesis been validated? A literature review," Current Issues in Tourism, vol. 19, no. 5, pp. 394-430, 2016.
[6] G. Wall, "From carrying capacity to overtourism: A perspective article," Tourism Review, vol. 75, no. 1, pp. 212-215, 2020.
[7] H. Goodwin, "The challenge of overtourism," Responsible tourism partnership, vol. 4, pp. 1-19, 2017.
[8] H. Moon and H. Han, "Tourist experience quality and loyalty to an island destination: The moderating impact of destination image," Journal of Travel Tourism Marketing, vol. 36, no. 1, pp. 43-59, 2019.
[9] V. Muler Gonzalez, L. Coromina, and N. Gali, "Overtourism: residents' perceptions of tourism impact as an indicator of resident social carrying capacity-case study of a Spanish heritage town," Tourism review, vol. 73, no. 3, pp. 277-296, 2018.
[10] I. A. Wong, Y. H. Xu, X. S. Tan, and H. Wen, "The boundary condition of travel satisfaction and the mediating role of destination image: The case of event tourism," Journal of Vacation Marketing, vol. 25, no. 2, pp. 207-224, 2019.
[11] F. Li, J. Wen, and T. Ying, "The influence of crisis on tourists' perceived destination image and revisit intention: An exploratory study of Chinese tourists to North Korea," Journal of destination marketing management, vol. 9, pp. 104-111, 2018.
[12] J.-C. Sánchez-Galiano, P. Martí-Ciriquián, and P. Fernández-Aracil, "Temporary population estimates of mass tourism destinations: The case of Benidorm," Tourism Management, vol. 62, pp. 234-240, 2017.
[13] J. Cramer and A. B. Krueger, "Disruptive change in the taxi business: The case of Uber," American Economic Review, vol. 106, no. 5, pp. 177-182, 2016.
[14] L. Abreu and A. J. Conway, "Invited Student Paper-A Qualitative Assessment of the Multimodal Passenger Transportation System Response to COVID-19 in New York City," 2021.
[15] X. Lu, E. Wetter, N. Bharti, A. J. Tatem, and L. Bengtsson, "Approaching the limit of predictability in human mobility," Scientific reports, vol. 3, no. 1, p. 2923, 2013..
[16] A. Cuttone, S. Lehmann, and M. C. González, "Understanding predictability and exploration in human mobility," EPJ Data Science, vol. 7, pp. 1-17, 2018.
[17] L. Chen, M. Lv, Q. Ye, G. Chen, and J. Woodward, "A personal route prediction system based on trajectory data mining," Information Sciences, vol. 181, no. 7, pp. 1264-1284, 2011
[18] M. Batty, "Predicting where we walk," Nature, vol. 388, no. 6637, pp. 19-20, 1997.
[19] X. Lu, L. Bengtsson, and P. Holme, "Predictability of population displacement after the 2010 Haiti earthquake," Proceedings of the National Academy of Sciences, vol. 109, no. 29, pp. 11576-11581, 2012.
[20] G. Athanasopoulos, R. J. Hyndman, H. Song, and D. C. Wu, "The tourism forecasting competition," International Journal of Forecasting, vol. 27, no. 3, pp. 822-844, 2011.
[21] A. Jungmittag, "Combination of forecasts across estimation windows: An application to air travel demand," Journal of Forecasting, vol. 35, no. 4, pp. 373-380, 2016.
[22] J. L. Chen, G. Li, D. C. Wu, and S. Shen, "Forecasting seasonal tourism demand using a multiseries structural time series method," Journal of Travel Research, vol. 58, no. 1, pp. 92-103, 2019.
[23] D. C. Wu, H. Song, and S. Shen, "New developments in tourism and hotel demand modeling and forecasting," International Journal of Contemporary Hospitality Management, vol. 29, no. 1, pp. 507-529, 2017.
[24] X. Huang, L. Zhang, and Y. Ding, "The Baidu Index: Uses in predicting tourism flows–A case study of the Forbidden City," Tourism management, vol. 58, pp. 301-306, 2017.
[25] C.-J. Lin, H.-F. Chen, and T.-S. Lee, "Forecasting tourism demand using time series, artificial neural networks and multivariate adaptive regression splines: Evidence from Taiwan," International Journal of Business Administration, vol. 2, no. 2, pp. 14-24, 2011.
[26] A. Aslanargun, M. Mammadov, B. Yazici, and S. Yolacan, "Comparison of ARIMA, neural networks and hybrid models in time series: tourist arrival forecasting," Journal of Statistical Computation and Simulation, vol. 77, no. 1, pp. 29-53, 2007.
[27] C.-F. Chen, M.-C. Lai, and C.-C. Yeh, "Forecasting tourism demand based on empirical mode decomposition and neural network," Knowledge-Based Systems, vol. 26, pp. 281-287, 2012.
[28] R. Girau, E. Ferrara, M. Pintor, M. Sole, and D. Giusto, "Be right Beach: A social IoT system for sustainable tourism based on beach overcrowding avoidance," in 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), 2018: IEEE, pp. 9-14.
[29] C. Kamga, M. A. Yazici, and A. Singhal, "Hailing in the rain: Temporal and weather-related variations in taxi ridership and taxi demand-supply equilibrium," in Transportation research board 92nd annual meeting, 2013, vol. 1.
[30] W. Jiang and L. Zhang, "Geospatial data to images: A deep-learning framework for traffic forecasting," Tsinghua Science and Technology, vol. 24, no. 1, pp. 52-64, 2018.
[31] S. Li, S. Bao, C. Yao, and L. Zhang, "Exploring the Spatio-Temporal and Behavioural Variations in Taxi Travel Based on Big Data during the COVID-19 Pandemic: A Case Study of New York City," Sustainability, vol. 14, no. 20, p. 13548, 2022.
[32] Taxi and Limousine Commission (TLC), "2019 Yellow Taxi Trip Data," City of New York, Updated: May 9, 2022, [Online]. Available: https://data.cityofnewyork.us/Transportation/2019-Yellow-Taxi-Trip-Data/2upf-qytp. [Accessed: Aug. 6, 2023].
[33] Taxi and Limousine Commission (TLC), "2018 Yellow Taxi Trip Data," City of New York, [Online]. Available: https://data.cityofnewyork.us/Transportation/2018-Yellow-Taxi-Trip-Data/t29m-gskq. [Accessed: Aug. 15, 2023].
[34] Taxi and Limousine Commission (TLC), "NYC Taxi Zones," City of New York, [Online]. Available: https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc. [Accessed: Aug. 8, 2023].
[35] OpenWeatherMap, "OpenWeatherMap," OpenWeatherMap. Available: https://openweathermap.org/. Accessed: Aug. 7, 2023.
[36] W. McKinney et al. "pandas - Python Data Analysis Library." pandas. https://pandas.pydata.org/ (accessed Aug. 15, 2023).
[37] Q. Huang et al., "NewYork-Busyness-Web," GitHub repository, [Online]. Available: https://github.com/QinghanHuang/NewYork-Busyness-Web.git. [Accessed: Aug. 6, 2023].
[38] Majeed, "Improving Time Complexity and Accuracy of the Machine Learning Algorithms Through Selection of Highly Weighted Top k Features from Complex Datasets," in Annals of Data Science, vol. 6, no. 3, pp. 599-621, Sep. 2019. https://doi.org/10.1007/s40745-019-00217-4.
[39] Enago Academy. "One-Hot Encoding: A Simple Yet Effective Way to Handle Categorical Data," in Enago Academy, Aug. 13, 2023. https://www.enago.com/academy/one-hot-encoding-a-simple-yet-effective-way-to-handle-categorical-data/. Accessed on: Aug. 6, 2023.
[40] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," arXiv preprint arXiv:2008.05756, 2020. https://arxiv.org/pdf/2008.05756. Accessed on: Aug. 6, 2023.