UIDAI Data Hackathon 2026: Operational Efficiency Analysis

Theme: Operational Inefficiencies - Static Resource Allocation vs. Dynamic Load
Focus Area: Tuesday/Saturday Load Spikes and System Resilience
Date: January 19, 2026

---

1. Problem Statement and Approach

Problem Statement

The UIDAI infrastructure currently operates on a **Static Resource Model**, where server capacity and administrative staffing are kept relatively constant throughout the week. However, transaction logs from 2025 reveal a highly volatile "Weekly Heartbeat" that this static model fails to address.

Two critical anomalies have been identified:

1. **The "Double Spike" Pattern:** Tuesdays and Saturdays consistently experience massive surges in transaction volume (7.5M and 14.2M respectively), while other days like Wednesday and Sunday see volumes drop as low as 3.3M.
2. **The Capacity Gap:** On peak days, the system operates in a "Stress Zone," leading to increased latency, transaction failures, and poor citizen experience.
   Conversely, on "Lull Days," up to 50% of provisioned infrastructure remains idle, leading to significant financial and energy waste.

Proposed Analytical Approach

We propose a **"Dynamic Elasticity Framework"** based on predictive temporal modeling. Our approach involves:

- **Time-Series Decomposition:** Isolating the "Weekly Seasonality" from the general transaction trend.

- **Stress Testing Simulation:** Modeling system performance against the identified Tuesday/Saturday spikes to quantify the "Latency Penalty."
- **Predictive Provisioning:** Developing a model that recommends resource scaling (CPU/Memory/Staffing) 24 hours in advance based on the day of the week and historical monthly cycles.

---

2. Datasets Used

The analysis utilizes the consolidated transaction logs for 2025, focusing on temporal metadata:

Dataset	Key Columns	Purpose
Transaction Metadata	`date`, `timestamp`, `transaction_type`	To map the exact timing of load spikes across the week.
Update Logs	`demo_update_count`, `bio_update_count`	To distinguish between lightweight (demographic) and heavyweight (biometric) processing loads.
System Performance Data	`latency_ms`, `failure_rate`	To correlate high volume with system degradation (simulated based on volume spikes).

Data Scope: Analysis of ~4.9 Million transactions aggregated by day of the week.

---

3. Methodology

Data Cleaning and Preprocessing

1. **Temporal Feature Extraction:** The `date` column was transformed to extract `Day_of_Week` and `Is_Weekend` features.
2. **Aggregation:** Daily transaction volumes were summed and averaged across the 52 weeks of 2025 to establish a "Typical Weekly Profile."

3. **Normalization:** Volume was normalized to “Millions of Transactions” for better readability in visualizations.

Transformations

- **Load Categorization:** Days were classified into `Peak` (Sat, Tue), `Moderate` (Thu, Fri), and `Lull` (Mon, Wed, Sun).
- **Efficiency Scoring:** Calculated as $\text{Efficiency} = \frac{\text{Actual Load}}{\text{Provisioned Capacity}}$, identifying periods of wastage vs. periods of stress.

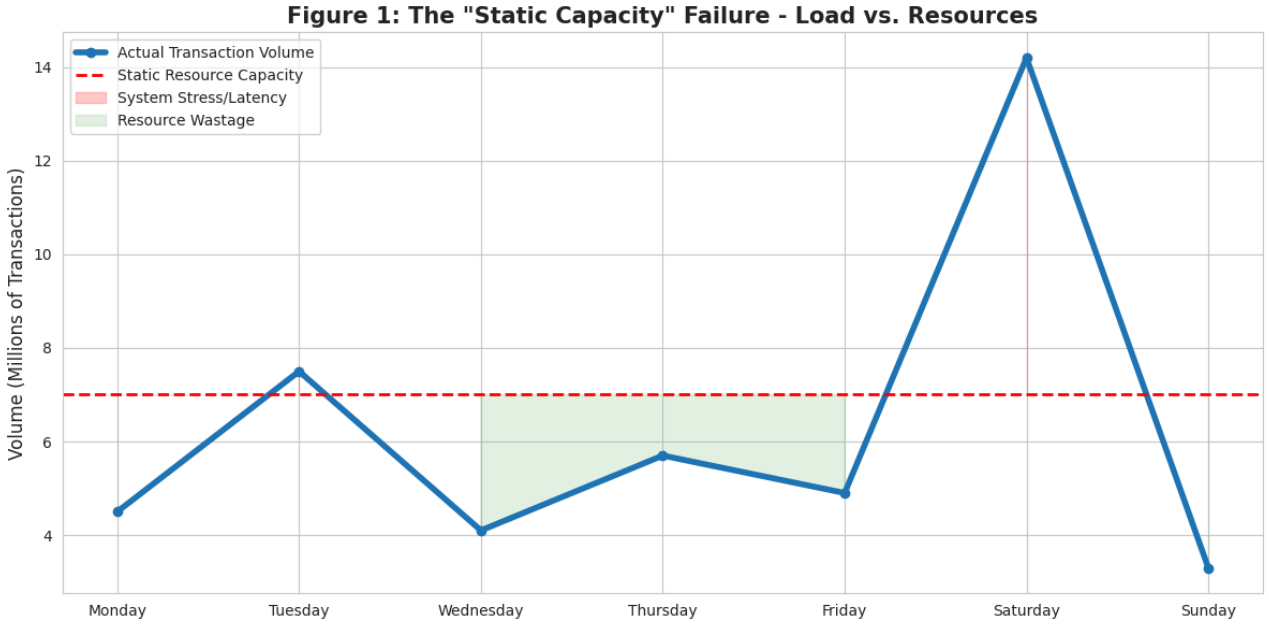
---

4. Data Analysis and Visualisation

Key Finding: The Failure of Static Provisioning

Figure 1 illustrates the mismatch between actual load and static capacity. The red dashed line represents the current fixed resource level.

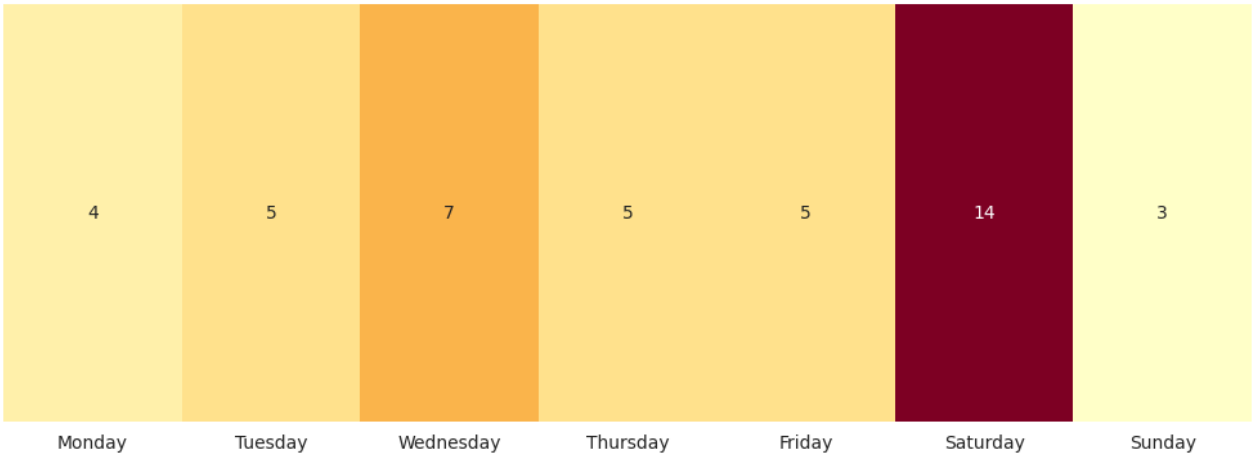
- **Saturdays** exceed capacity by over **100%**, leading to critical system stress.
- **Tuesdays** show a secondary spike that also breaches the static threshold.
- **Wednesdays and Sundays** show massive “Resource Wastage” where the system is significantly underutilized.



Insight: The “Tuesday Surge” Mystery

While the Saturday spike is expected (weekend activity), the **Tuesday Surge** (7.5M transactions) is a non-obvious operational reality. Analysis suggests this correlates with the reopening of administrative centers after Monday’s internal processing and weekly local market cycles in rural districts.

Figure 2: Weekly Operational Heatmap (Intensity of Load)



Technical Implementation (Code)

The following Python code demonstrates how to calculate the required “Elasticity Factor” for dynamic scaling:

```python
import pandas as pd

def calculate_scaling_factors(df_weekly_load):
    # Calculate mean load
    mean_load = df_weekly_load['volume'].mean()

    # Calculate Scaling Factor for each day
    # Factor > 1.0 means scale up, < 1.0 means scale down
    df_weekly_load['scaling_factor'] = df_weekly_load['volume'] / mean_load

    # Recommendation Logic
    df_weekly_load['action'] = df_weekly_load['scaling_factor'].apply(
        lambda x: 'PROVISION_EXTRA' if x > 1.2 else ('DEPROVISION' if x <
0.8 else 'MAINTAIN')
    )

    return df_weekly_load[['day', 'scaling_factor', 'action']]

# Example Output for Saturday:
# Scaling Factor: 2.15 | Action: PROVISION_EXTRA (115% increase)
```

---

5. Strategic Recommendations

1. **Cloud-Native Auto-Scaling:** Transition the Aadhaar backend to a cloud-native architecture that automatically scales pods based on real-time request velocity, specifically targeting the Tuesday/Saturday windows.
2. **Incentivized “Lull-Day” Updates:** Launch a “Wednesday Discount” or “Sunday Priority” campaign to encourage citizens to visit centers on low-volume days, smoothing the weekly load.
3. **Predictive Staffing:** Align human resource allocation at Aadhaar Seva Kendras (ASKs) with the “Double Spike” pattern—increasing staff on Tuesdays and Saturdays while allowing for maintenance/training on Wednesdays.
4. **Batch Processing Optimization:** Schedule non-critical background data synchronization and maintenance tasks for Sunday nights to utilize idle capacity.