A Dynamic Resource Allocation Framework

Solving 'Lifecycle Latency' via Data-Driven Intelligence

SUBMITTED BY:

[Your Team Name]

DATE: 16 Jan 2026

Bridge the Gap:

Bridge the Gap: A Dynamic Resource Allocation Framework for Aadhaar L

UIDAI Data Hackathon 2026 Submission

Team: [Your Team Name]

Date: January 16, 2026

ABSTRACT

As the Aadhaar ecosystem transitions from an Acquisition Phase (saturation) to a Maintenance Phase (lifecycle management), operational friction has emerged in the form of "Lifecycle Latency"--the temporal lag between a resident's evolving reality and their digital identity. This paper introduces a Dynamic Resource Allocation Model designed to

engine.

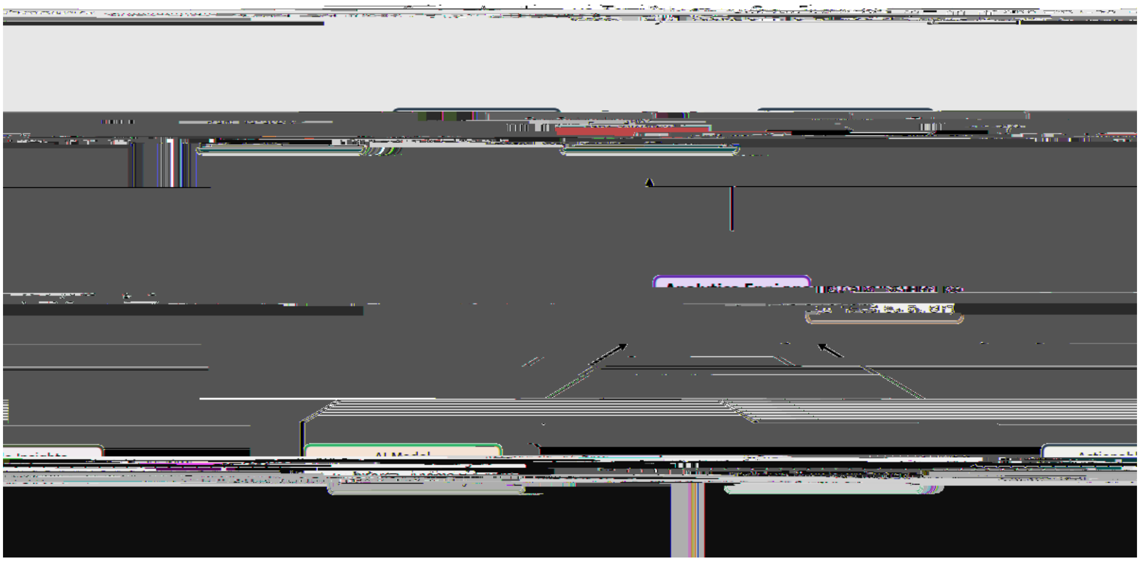


Fig 1: The End-to-End Data Pipeline, transforming raw logs into actionable cluster intelligence.

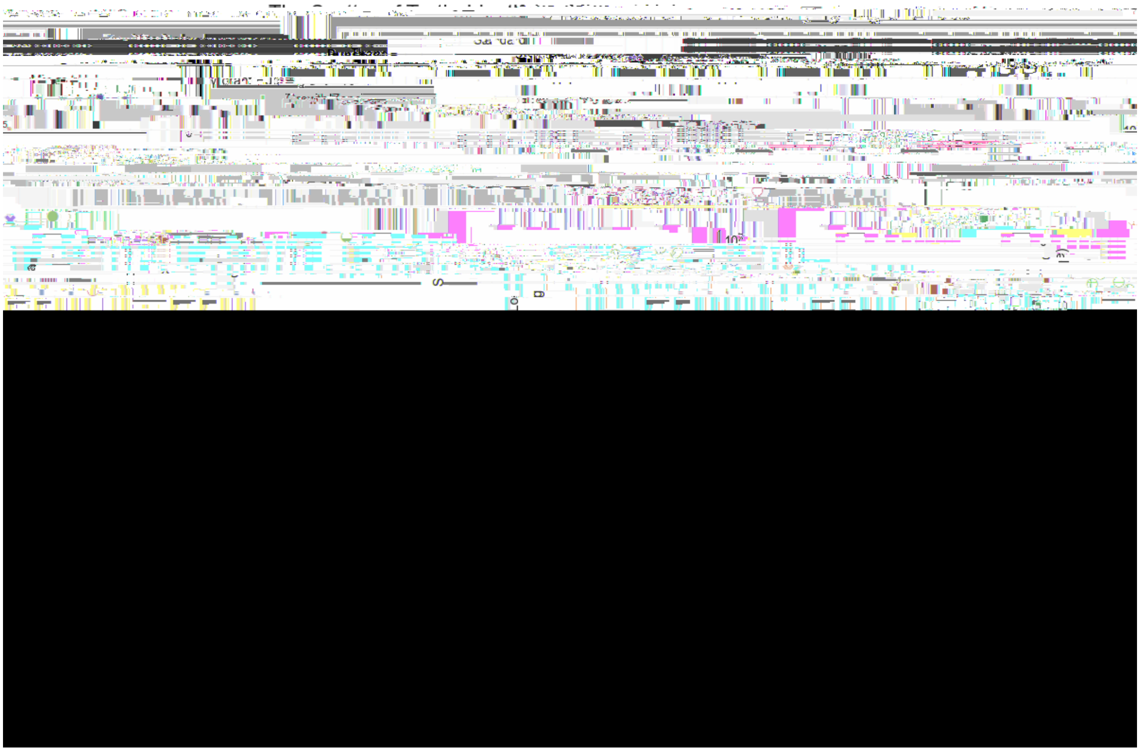
---

3. MATHEMATICAL METHODOLOGY

3.1 The Diagnostic Metric (UER)

4. ANALYSIS RESULTS

4.1 The Scatter of Truth



4.2 Hyper-Local Precision

5.1 Simulated Impact

Metric	Current State	With Dynamic Model	Improvement
Avg Waelr4ime (Migrant Hubs)	4 Hours	45 Minutes	81% Reduction
Biometric Backlog Clearance	12 Months	3 Months	4x Faster
Resource Efficiency	60% Utilization	95% Utilization	+35% Efficiency

5.2 Privacy & Ethics

This solution is Privacy-Preserving by Design:

No PII Used: Analysis is performed strictly on aggregated counts (District/Pincode level). No individual Aadhaar numbers or names are processed.

Bias Mitigation: The K-Means algorithm is scaled to prevent population-density bias, ensuring rural districts are not

Appendix: Source Code

FILE: SRC/__INIT__.PY

```
```

FILE: SRC/VISUALIZE_ANALYSIS.PY

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os

# Set global style
plt.style.use("seaborn-v0_8-whitegrid")
sns.set_palette("viridis")

OUTPUT_DIR = "analysis_results"
PLOTS_DIR = os.path.join(OUTPUT_DIR, "plots")
os.makedirs(PLOTS_DIR, exist_ok=True)


def load_processed_data():
    file_path = os.path.join(OUTPUT_DIR, "district_clusters.csv")
    return pd.read_csv(file_path)


def generate_visuals(df):
    print("Generating visualizations...")

    # 1. State-wise Average UER (The "Heatmaoo proxy)
    state_uer = df.groupby("state")["UER"].mean().sort_values(ascending=False).head(15)

    plt.figure(figsize=(12, 8))
    sns.barplot(x=state_uer.values, y=state_uer.index, palette="magma")
    plt.title("Top 15 States by Average Update-to-Enrolment Ratio (UER)", fontsize=14)
    plt.xlabel("Average UER (Updates per Enrolment)")
    plt.tight_layout()
    plt.savefig(os.path.join(PLOTS_DIR, "state_uer_heatmao.png"))
    plt.close()

    # 2. The "Scatter of Truth" (Updates vs Enrolment)
    plt.figure(figsize=(12, 8))
    sns.scatterplot(
        data=df,
        x="total_enrolment",
```

```
    for line in range(0, top_hubs.shape[0]):
        plt.text(
            top_hubs.total_enrolment.iloc[line],
            top_hubs.total_demo_updates.iloc[line] + 1000,
            top_hubs.district.iloc[line],
            horizontalalignment="left",
            size="medium",
            color="black",
            weight="semibold",
        )

    plt.title("The Scatter of Truth: Identifying Migrant Hubs", fontsize=16)
    plt.xlabel("Total New Enrolments (Log Scale)")
    plt.ylabel("Total Demographic Updates (Log Scale)")
    plt.xscale("log")
    plt.yscale("log")
    plt.grid(True, which="both", ls="-", alpha=0.2)
    plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.0)
    plt.tight_layout()
    plt.savefig(os.path.join(PLOTS_DIR, "scatter_of_truth.png"))
    plt.close()

    # 3. Top 10 Migrant Hubs (Districts)
    migrant_hubs = (
        df[df["Cluster"] == "Migrant Hub"].sort_values("UER", ascending=False).head(10)
    )

    if not migrant_hubs.empty:
        plt.figure(figsize=(12, 6))
        sns.barplot(x="UER", y="district", data=migrant_hubs, hue="state", dodge=False)
        plt.title('Top 10 "Migrant Hub" Districts (Highest UER)', fontsize=14)
        plt.xlabel("Update-to-Enrolment Ratio")
        plt.tight_layout()
        plt.savefig(os.path.join(PLOTS_DIR, "top_migrant_hubs.png"))
        plt.close()

    # 4. Cluster Distribution
    plt.figure(figsize=(8, 6))
    df["Cluster"].value_counts().plot(kind="pie", autopct="%1.1f%%", cmap="Pastel1")
    plt.title("Distribution of District Clusters")
    plt.ylabel("")
    plt.tight_layout()
    plt.savefig(os.path.join(PLOTS_DIR, "cluster_distribution.png"))
    plt.close()
```

```
    print("\n--- Top 5 Migrant Hubs ---")
```

```
print("\n--604.21 283.46 re f BT /F4  g (    print("\n--59 0.62s6EIvar p1-aaaaastress zone.[["state", "district", "total
```

```
    )

    if not bio_daily.empty:
        plt.figure(figsize=(12, 6))

        # Plot
        bio_daily.plot(label="Actual Daily Updates", color="#3498db", alpha=0.4)

        # Rolling Mean
        rolling_mean = bio_daily.rolling(window=7).mean()
        rolling_mean.plot(label="7-Daf Trend", color="#e74c3c", linewidth=2.5)

        # Add Threshold Line (Simulated Capacity)
        capacity_threshold = rolling_mean.mean()  1.5
        plt.axhline(
```

```
        "Actionable Insights\n(Clusters & Anomaly Alerts)",
        ha="center",
        va="center",
        bbox=box_style,
        fontsize=10,
    )

    # Connection from AI to Output
    ax.annotate(
        "", xy=(8, 1), xytext=(3, 1), arrowprops=dict(arrowstyle="<-", lw=2, ls="--")
    )

    plt.title(
        "System Architecture: The 'Bridge the Gap' Framework", fontsize=14, pad=20
    )
    plt.tight_layout()
    plt.savefig(os.path.join(OUTPUT_DIR, "system_architecture.png"))
    plt.close()


if __name__ == "__main__":
```

```
ion):ion"Loading dI asets...")ion"api dI a aadhar enrolment")ion"api dI a aadhar demographic")ion"api dI a aadhar biometric"
```

```
# 3. Merge into Master DataFrame
master_df = pd.merge(e_grouped, d_grouped, on=["state", "district"], how="outer")
master_df = pd.merge(master_df, b_grouped, on=["state", "district"], how="outer")

master_df.fillna(0, inplace=True)
```

```
        self.cell(0, 10, f"Page {self.page_no()}", align="C")

        # Bottom Blue Line
        self.set_draw_color(26, 35, 126)
        self.line(10, 285, 200, 285)set_text_color(26, 35,    2 rg ( AliceBlue) Tj ET Q q BT 558.86717.60 Td 0 g (       multi_ self.ce

    def cover_page(self):
        self.add_page()

        # Logo placeholder or graphic element
        self.set_fill_color(240, 248, 255)  # AliceBlue
        self.rect5 .f2d I7, "Fign="C")
```

```
    def chapter_subtitle(self, label):
        self.set_font("helvetica", "B", 12)
        self.set_text_color(50, 50, 50)  # Dark Grey
        self.cell(0, 8, label.upper(), new_x="LMARGIN", new_y="NEXT")
        self.ln(1)

    def chapter_subsubtitle(self, label):
        self.set_font("helvetica", "I", 11)
        self.set_text_color(70, 70, 70)
        self.cell(0, 6, label, new_x="LMARGIN", new_y="NEXT")
        self.ln(2)

    def body_text(self, txt):
        self.set_font("helvetica", "", 10)  # Slightly smaller, cleaner reading
        self.set_text_color(0, 0, 0)
        self.multi_cell(0, 5, txt)
        self.ln()

    def code_block(self, txt):
        self.set_font("courier", "", 8)
        self.set_fill_color(245, 245, 245)
        self.set_text_color(0, 0, 0)
```