StatsAPI - TECHNICAL REPORT

A Lightweight GitHub Hosted Metrics Endpoint for Distributed UI Consumers

Author: Amjad Ali Kudsi **Date:** November 20, 2025

Problem Overview

Systems that visualize user progress often require frequent updates to numerical indicators, such as completed projects or solved coding problems. When these values are hard coded within a portfolio application, every update forces a full rebuild and deployment of the site. Additionally, platforms such as CodeSignal and LeetCode do not expose official public APIs to retrieve these values programmatically.

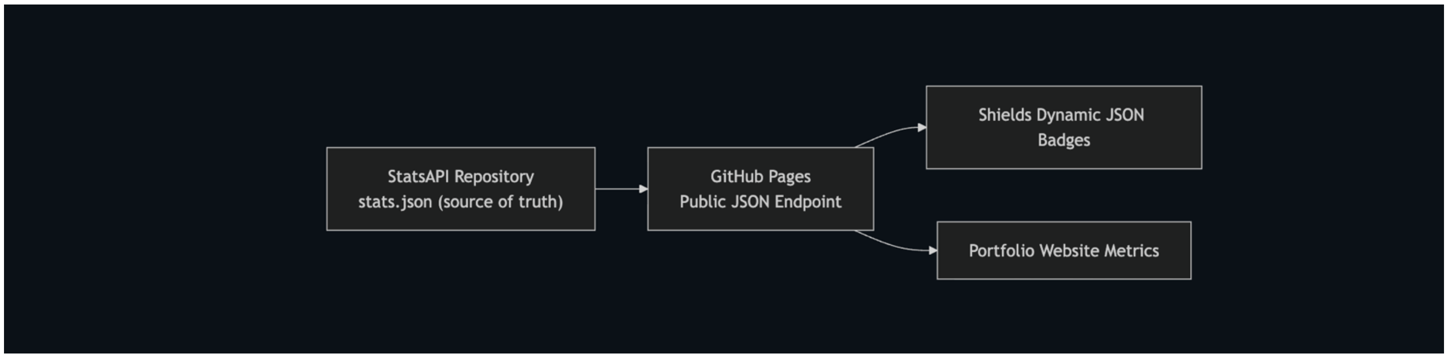
StatsAPI introduces a controlled, versioned, and externally accessible endpoint that exposes these metrics as a simple JSON document. By decoupling the data layer from the portfolio application, the system enables consistent and low-friction updates without requiring site redeployment.

Technical Architecture

StatsAPI is implemented as a GitHub repository that hosts a structured JSON document and exposes it through GitHub Pages. This repository serves as the authoritative data layer for all downstream consumers.

Components:

- **stats.json**: A structured JSON document acting as the single source of truth for all metrics.
- **GitHub Pages**: Provides a stable, publicly accessible HTTPS endpoint for ‘stats.json’.
- **Consumers**:
    - Shields dynamic JSON badges incorporated into the GitHub profile.
    - A portfolio application that fetches and displays the same metrics.



Figma

Design Rationale

◊ Single Source of Truth

Consolidating all metrics into one JSON document prevents data divergence and provides a canonical reference for all consuming systems. JSON is used due to its widespread interoperability, suitability for static hosting, and native compatibility with client-side rendering environments and Shields' JSON badge queries.

◊ GitHub Pages as the Delivery Mechanism

GitHub Pages was selected as the hosting medium due to the following technical properties:

- automatic deployment on commit
- public static content delivery
- predictable HTTPS endpoint
- integration with Git versioning

These attributes collectively enable a compact and reliable mechanism for distributing machine readable data without provisioning backend infrastructure.

◊ Shields Dynamic JSON Badges as Validation Layer

Shields dynamic JSON badges read directly from the StatsAPI endpoint. Their purpose extends beyond visual display:

- they provide immediate confirmation that the JSON endpoint is updated
- they eliminate the need to check the portfolio application to verify updates
- they act as a passive monitoring surface integrated into the GitHub profile

This results in a minimal yet effective validation strategy.

◊ Removal of Redundant Components

Earlier iterations incorporated:

- hidden embedded JSON inside the GitHub profile README
- cross repository synchronization pipelines
- fine grained personal access tokens
- GitHub Actions workflows propagating data across repositories

These were functional but unnecessary for the core requirement. Eliminating them led to:

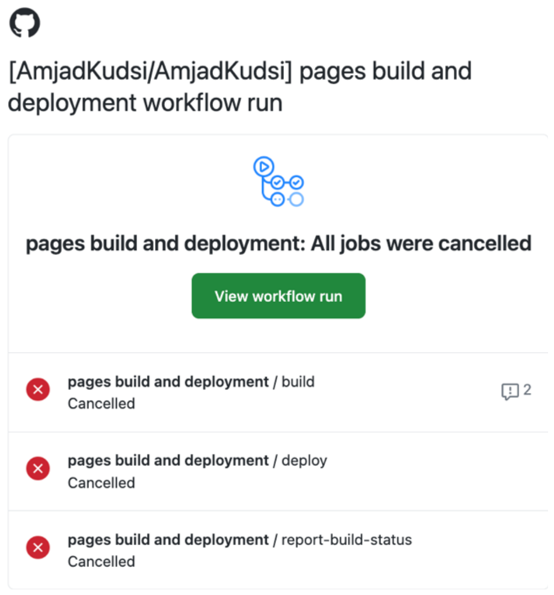
- simplified architecture with a single repository

- removal of workflow complexity
- elimination of cross repo write operations
- consistent one commit per update cycle
- fully deterministic GitHub Pages deployment behavior

This refinement significantly improved maintainability and efficiency.

Architecture Evolution Summary

Version	Description	Reason for Change
V1	Metrics hard coded in the portfolio	Required full redeployment for every update
V2	Hidden JSON block inside GitHub profile README with workflow extraction	Introduced unnecessary workflow activity and deployment noise
V3	Two repository architecture with cross repo sync and PAT	Provided separation but exceeded the actual problem scope
V4 (StatsAPI)	Single repository serving static JSON via GitHub Pages	Most efficient architecture fulfilling all functional requirements


[AmjadKudsi/AmjadKudsi] pages build and deployment workflow run

pages build and deployment: All jobs were cancelled

pages build and deployment / build
Cancelled

pages build and deployment / deploy
Cancelled

pages build and deployment / report-build-status
Cancelled

Email notification - cancelled jobs

Operational Insights

- Minimal architectures that directly satisfy functional requirements tend to exhibit higher clarity and reliability.
- Static hosting solutions such as GitHub Pages provide effective infrastructure for serving lightweight machine-readable data.
- Dynamic JSON badges offer a practical mechanism for passive validation and status visibility.
- Cross repository workflows and token-based authentication are powerful capabilities but should be employed only when justified by system complexity.
- Maintaining a single authoritative data source prevents drift and reduces operational overhead.

Possible Future Extensions

Potential enhancements include:

- Automated population of metrics through official APIs if they become available
- Scheduled or event driven updates to 'stats.json'
- Historical data logging for analysis and visualization
- Additional endpoints supporting derived or aggregated metrics