Paper DV09

Visualizing Trends and Bottlenecks: A Python Web App for QC Progress Monitoring

Spencer Renyard, Phastar, London, United Kingdom

ABSTRACT

Tracking the progress of QC efforts across multiple programs can be challenging without clear, actionable visualizations. To address this, we developed a Python® web application that accesses QC tracker data via the Monday.com API, using both live data and the activity log to determine the status of each program over time and when changes occurred. The application generates graphical representations of progress, including a weighted percentage based on the number of programs in each status category.

By automating the extraction and analysis of QC tracker data, the application reduced manual effort and provided a historical view of program progress. Its visualizations enabled teams to identify trends, track bottlenecks, and make data-driven decisions, improving oversight of QC activities.

This paper will include a technical walkthrough of the application's development, covering Python libraries, the Monday.com API, and visualization techniques.

INTRODUCTION

Quality Control (QC) tracking is a vital component of clinical programming workflows and has traditionally been managed using Excel spreadsheets. While Microsoft Excel® offers familiarity and flexibility, it suffers from limitations in collaborative environments such as Microsoft SharePoint®, lacks historical tracking capabilities, and requires manual updates that are prone to error. This project was initiated with a fundamental question:

What if we could move beyond Excel?

And then, what if we could visualize not only the current status of QC programs but also their historical progression, identify trends, and automate the tracking process?

To explore this possibility, a Python-based web application was developed that integrates with Monday.com – a platform already in use within the organization – to access QC tracker data and activity logs. The application provides interactive, time-based visualizations that reflect both real-time and historical QC status, thereby enabling improved decision-making and operational oversight.

BACKGROUND AND MOTIVATION

BACKGROUND

Historically, QC tracking has relied heavily on Excel spreadsheets.

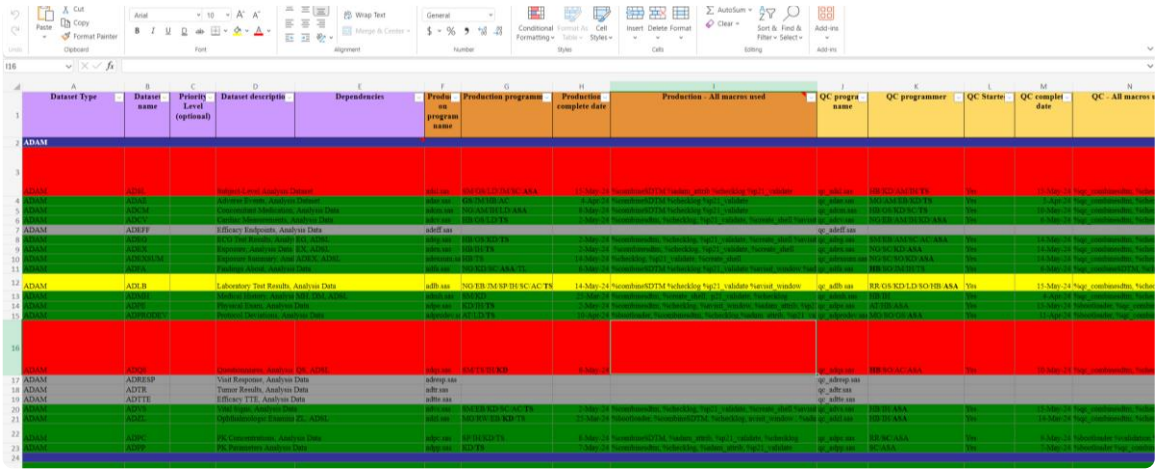

Figure 1 Example of an Excel QC tracker

Despite its widespread use and accessibility, Excel presents several limitations, including providing only static snapshots of program status, and it lacks the historical context necessary for trend analysis.

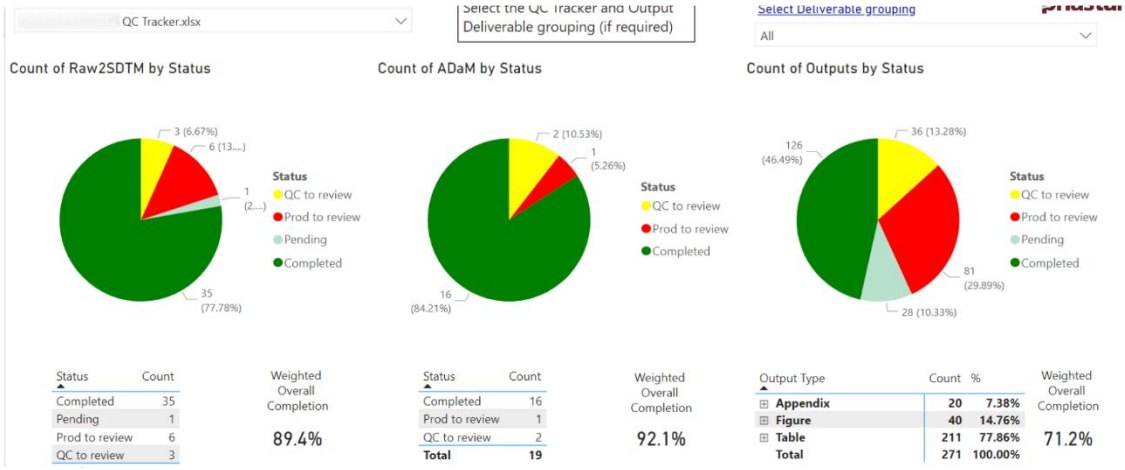

Figure 2 Example snapshot of QC tracker program status

Status changes and timestamps must be manually entered, increasing the risk of errors. Collaboration is hindered by versioning issues and incompatibility with macros and VBA, especially when files are stored on platforms like SharePoint. Furthermore, Excel lacks a robust audit trail, making it difficult to track changes and maintain transparency.

MOTIVATION

These limitations highlighted the need for a more robust, automated, and collaborative solution. The objective was to build a system capable of tracking QC progress over time, automatically logging changes and user actions, and providing visual insights into trends and bottlenecks. Ultimately, the goal was to enhance operational efficiency across globally distributed teams.

SYSTEM DESIGN AND ARCHITECTURE

The proposed solution is a Python-based web application that integrates seamlessly with Monday.com. The application accesses two primary data sources: the board data, which reflects the current status of each program:

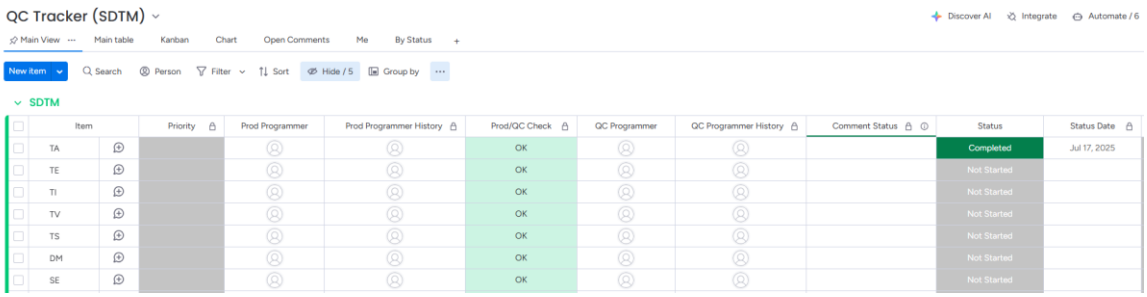

Figure 3 Example QC tracker board

And the activity log, which provides a historical record of status changes and user actions:

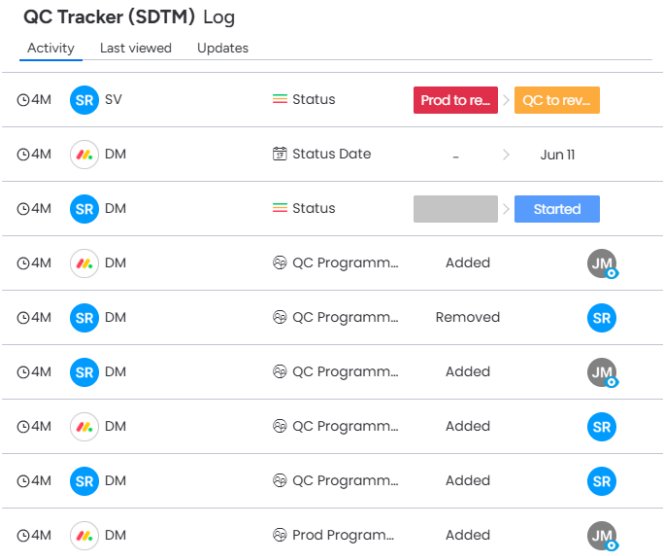

Figure 4 Example of the activity log

Although Monday.com offers a point-and-click interface with useful features such as drilldown and hover-over capabilities, it still only presents the current status of programs.

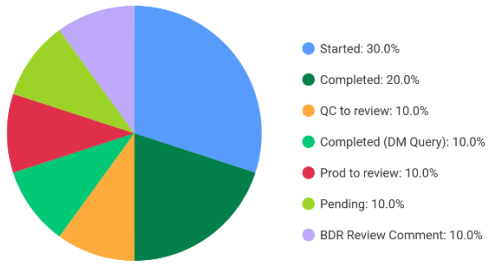

Figure 5 Example graphical object containing a pie chart showing the current status of programs

The developed application addresses this limitation by offering historical insights.

PROJECT OBJECTIVES

The overarching objective of the project is to create a visual representation of overall status for the duration of the study, tracking progress over time.

- **Scope definition**: What can be done with the data that is now available and how to report it
- **Data**: Extract and combine the QC tracker and activity log data to get every time the status changes for each program
- **Aggregation**: Calculate the weighted percentage complete per day based on the status value for each program
- **Annotations**: Ability to add annotations to the visualisation allowing milestones like Database Lock to be presented
- **Visualisation**: Visual representation of overall status for the duration of the study for each program type: Raw, SDTM, ADaM, Outputs

SYSTEM ARCHITECTURE

The backend is built using the Flask web framework, while the frontend leverages Plotly® to deliver interactive visualizations. Data is retrieved via Monday.com's GraphQL API, allowing for efficient querying of both board and activity log data. User authentication is managed through API keys, ensuring that access is appropriately scoped to individual permissions. To enhance performance, the application aggregates data to calculate a weighted percentage of completion per day, prioritizing completed programs over those still in progress. The technical stack includes Visual Studio Code for development, Docker® for containerization, GitLab® for version control and CI/CD workflows, Flask-SocketIO for real-time updates, and caching strategies to optimize API calls and minimize latency.

DATA AGGREGATION AND PROCESSING

Having the current and previous status values and dates available from the data in the QC tracker board and the activity log, this can be joined together and the status on any given day during the study determined.

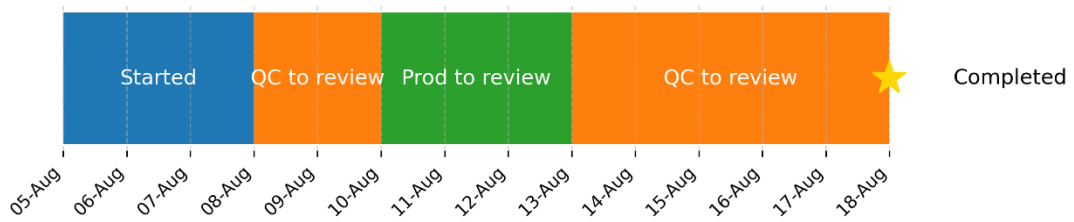

Figure 6 Program status over time

WEIGHTED PERCENTAGE OF COMPLETION PER DAY

Each value of Status is assigned a weight:

```
# Define status weights for all other program types
default_status_weights = {
    'Completed': 1.0,
    'Completed (DM Query)': 1.0,
    'Pending': 0.75,
    'QC to review': 0.5,
    'Prod to review': 0.5,
    'BDR Review Comment': 0.5,
    'Started': 0.25,
    'Not started': 0.0
}
```

The weighted status values for each day are summed up and using the overall number of programs as the denominator the Weighted Status Percentage for each day is calculated.

VIZUALIZATION TECHNIQUES

The core output of the application is an interactive line plot that illustrates QC progress over time. A stepped line chart is used to reflect discrete status changes, avoiding misleading interpolation. The weighted percentage of completion is calculated based on the number of programs in each status category.

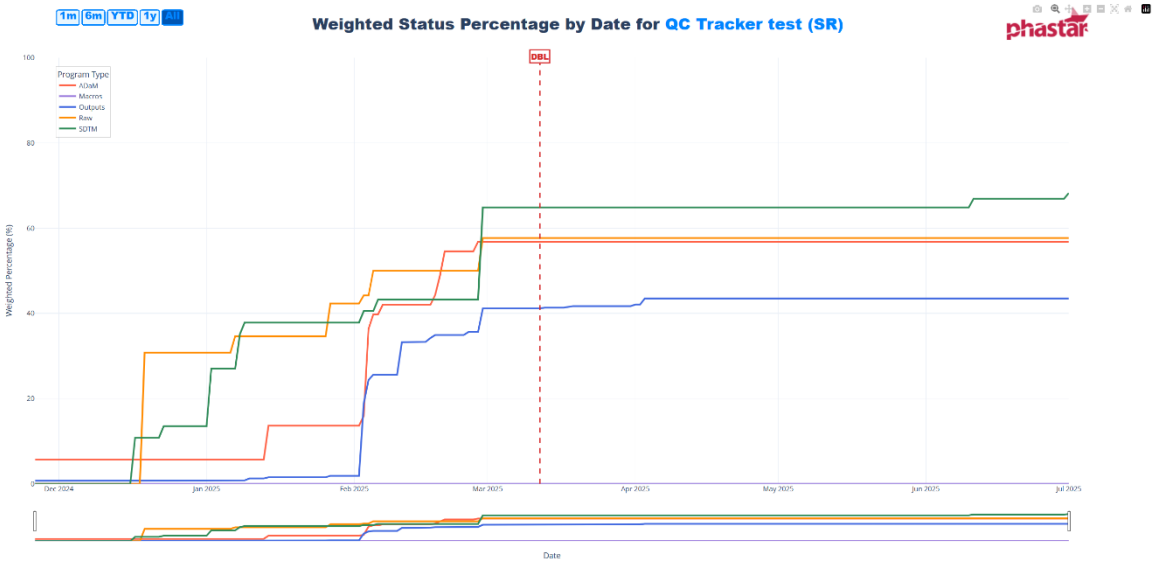

Figure 7 Example of the plot presenting weighted status percentage over time

This visualization incorporates several key features.

Milestone events, such as Database Lock (DBL), are annotated with vertical lines to provide contextual markers, as shown by the dashed red vertical line and “DBL” annotation above.

Users can utilize a date slicer to zoom into specific time periods:

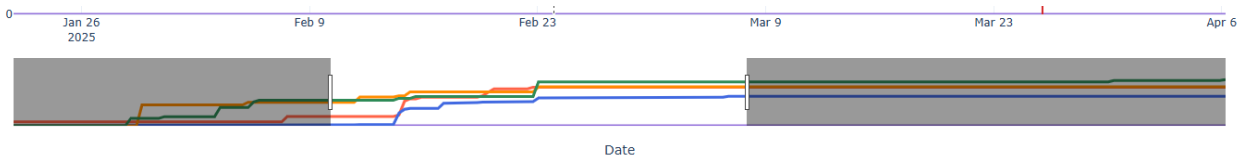

Figure 8 Date slicer

The plot includes hover-over tooltips to display detailed information for each data point:

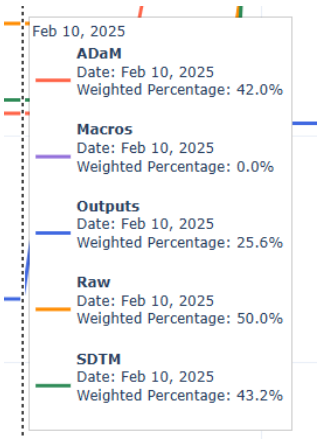

Figure 9 Hover over tooltip showing the date and weighted percentage

Program type filtering allows users to isolate a specific program category such as SDTM with a double-click on that program category in the legend:

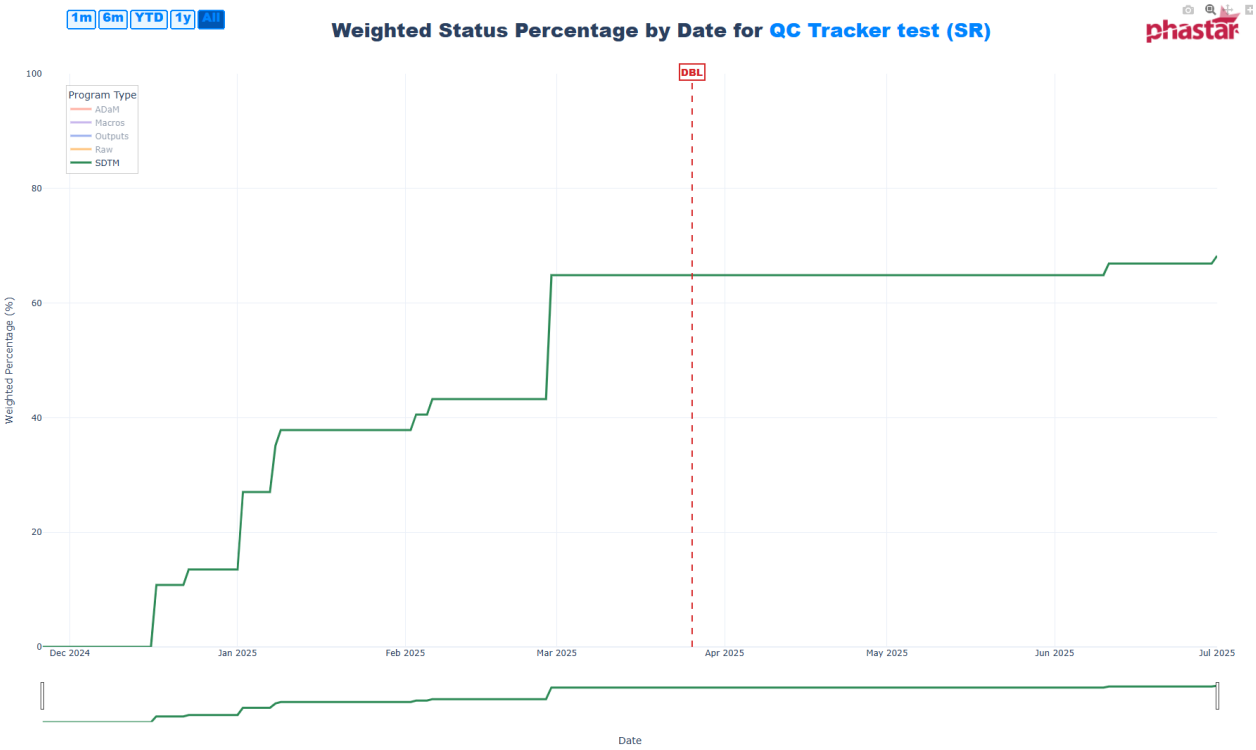

Figure 10 Selecting one program type to display with a double-click in the legend

Or a single-click will remove the program category from the plot, for example, to focus on the SDTM and ADaM programs:

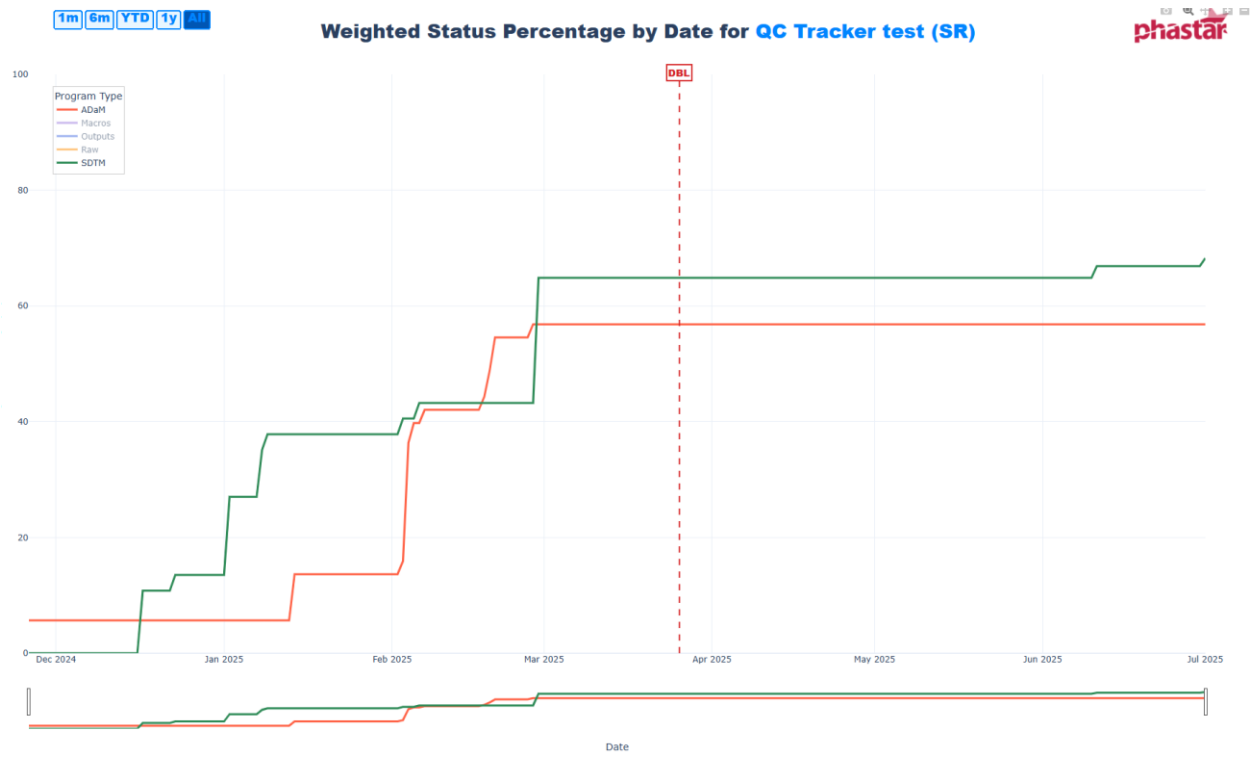

Figure 11 Selecting multiple program types to display

These visualizations offer a clear and actionable view of QC progress, enabling teams to identify trends, monitor bottlenecks, and make informed decisions.

DEVELOPMENT CHALLENGES

The development of the application involved overcoming several technical and conceptual challenges. Transitioning from SAS® to Python, Flask, and GraphQL required significant upskilling. Designing a user-friendly web interface and managing asynchronous data loading posed additional complexities. Mastery of GraphQL was essential to understand query structures and process JSON responses effectively.

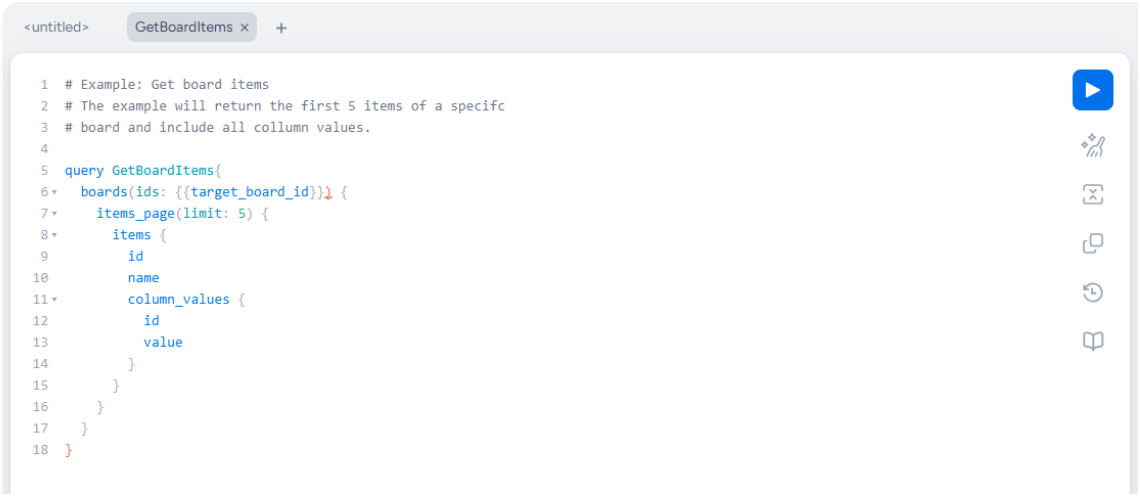

Figure 12 Example GraphQL query

Robust testing frameworks were implemented to validate functionality through unit and integration tests. Containerization using Docker facilitated deployment on internal servers, while GitLab workflows supported

collaborative development and CI/CD automation. A fundamental mindset shift was necessary to move from a SAS®-centric approach to a modern, service-oriented architecture.

RESULTS AND IMPACT

The application has delivered several tangible benefits. Automation has eliminated manual data entry and chart creation. Historical tracking capabilities have enabled visibility into program status changes over time. Collaboration has improved through multi-user access and reduced file locking issues. Data integrity has been enhanced via automated timestamping and audit trails. The system supports global teams with performant, region-aware access, thereby improving scalability.

Overall, the application has provided significant business value by enhancing oversight and decision-making capabilities. It has proven to be a valuable tool for QC tracking, offering insights that were previously inaccessible through Excel-based methods.

CONCLUSION

The development of a Python-based web application for QC progress monitoring represents a significant advancement over traditional Excel-based tracking systems. By integrating with Monday.com's API and leveraging both board data and activity logs, the application provides a dynamic and historical view of QC status across multiple programs. This enables teams to identify trends, monitor bottlenecks, and make informed decisions with greater confidence. The use of stepped line charts, weighted percentage calculations, and milestone annotations has transformed QC tracking from a static snapshot into a rich, interactive experience. The system's architecture, built with Flask, Plotly, Docker, and GitLab, supports scalability, automation, and secure access control, making it suitable for deployment across global teams.

Future enhancements include the development of advanced dashboards incorporating burn-up/burn-down charts, resource utilization metrics, and predictive analytics. UI/UX improvements will focus on refining the interface for better usability and accessibility across devices. Optimization of API calls will aim to balance data volume and frequency to improve performance and reduce latency. Plans for production deployment include transitioning from a development version to a fully supported environment with robust monitoring and support. As development expertise matures, architectural decisions will be revisited to ensure long-term maintainability and value, while evaluating the impact of emerging technologies and frameworks.

RECOMMENDED READING

Plotly Open Source Graphing Library for Python (https://plotly.com/python/)

Plotly GitHub (https://github.com/plotly/plotly.py)

Introduction to GraphQL (https://graphql.org/learn/)

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Author Name: Spencer Renyard
Company: Phastar
Address: Unit 2, 2D Bollo Lane, Chiswick, London, W4 5LE
Email: spencer.renyard@phastar.com
Website: phastar.com