VEAT System Architecture Documentation

1. Introduction

This document outlines the current and proposed system architecture for VEAT, a fast-growing foodtech company. The primary objective is to introduce a new **Plater Time Tracking Application** to address inefficiencies in tracking plating times, cost analysis, and basic scheduling within the kitchen operations. This application aims to replace the current manual system involving punch clocks and Google Sheets, providing a more efficient and accurate solution for managing production workflows.

2. Current VEAT Infrastructure Overview

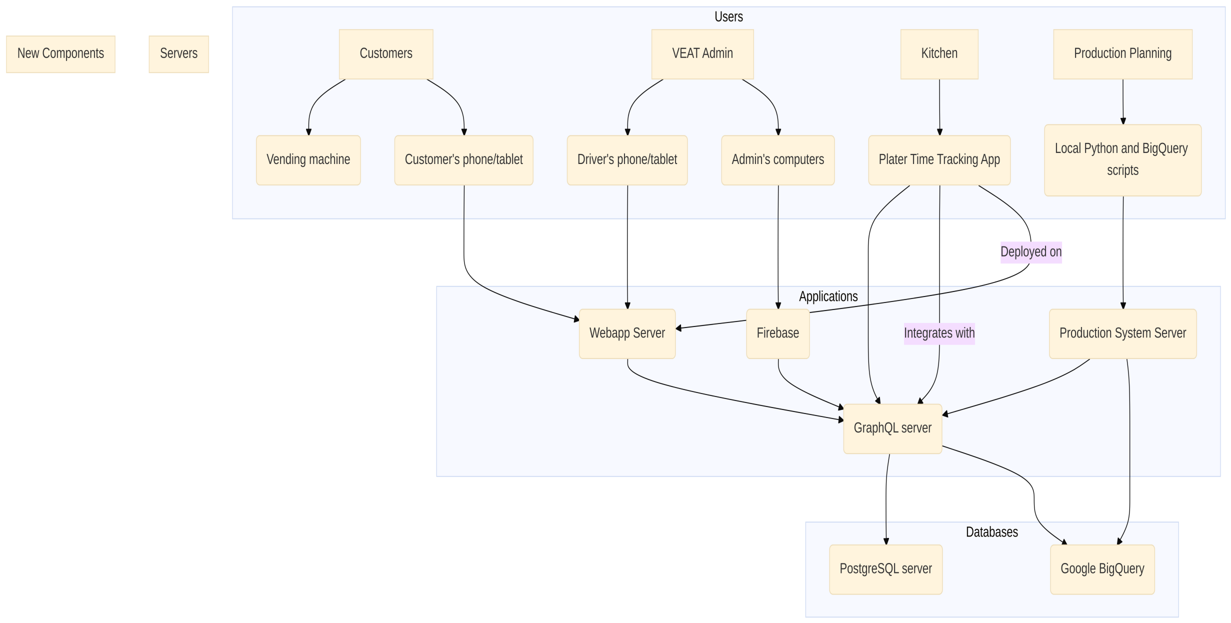
VEAT's existing infrastructure supports various user roles and operational needs, as depicted in the original system overview. Key components include:

- **Customers**: Interact with Vending machines and customer-facing phone/tablet applications.
- **VEAT Admin**: Utilizes driver's phone/tablet and admin's computers for administrative tasks.
- **Kitchen**: Currently uses manual methods for tracking, but interacts with other systems indirectly.
- **Production Planning**: Relies on local Python and BigQuery scripts.
- **Webapp Server**: Serves customer and driver applications.
- **Firebase**: Likely used for real-time data or authentication, connected to Admin's computers.
- **Production System Server**: Supports production planning scripts.
- **GraphQL Server**: Acts as an API gateway, connecting various applications to backend services.
- **PostgreSQL Server**: The primary relational database.
- **Google BigQuery**: Used for large-scale data analytics, particularly for production planning.

The core of the existing system appears to be the **GraphQL server**, which interfaces with both the **PostgreSQL server** and **Google BigQuery**, suggesting a centralized data access layer. The **Webapp Server** handles interactions from customer and driver applications, while the **Production System Server** supports internal production processes.

3. Proposed System Architecture

The proposed system architecture introduces a new **Plater Time Tracking Application** designed to integrate seamlessly with the existing VEAT infrastructure. This application will be a lightweight internal tool, preferably built using a low-code platform like Retool, to ensure rapid development and maintainability. The updated architecture, including the new application, is illustrated in the diagram below



3.1. Plater Time Tracking Application

This new application will serve as the central hub for kitchen staff to manage and track their work. Its key features include:

MVP (Must-Haves)

- **Plater Time Tracking**:
  - Individual logins for Platers.
  - Ability to select SKU (GUID from product table via GraphQL) or non-SKU tasks (e.g., dishwashing, packing).
  - Start/Stop timers, optimized for mobile/tablet use.
  - Support for multiple workers on the same SKU concurrently.
  - Optional notes for anomalies.
- **Admin Functions**:
  - Add/edit SKUs, steps, and tasks (configurable).
  - Set/update efficiency targets per SKU and/or step.
- **One-off Data Import**:
  - Semi-manual import of historical data from Google Sheets/CSV.
- **In-App Analytics & Dashboards**:
  - Display planned vs. actual time, weekly trends by SKU, step, and Plater.
  - Calculate labor cost per SKU, including allocation of non-SKU overhead tasks.

    - Export data to CSV/Excel.
    - Optional reconciliation view for cross-checking hours with external systems (e.g., Personalkollen).
- **Basic Scheduling/Benchmarking**:
    - Simple view of estimated hours needed per day based on historic timings and targets.

Phase 2 (Nice-to-Haves)

- Simple Plater dashboard with progress vs. efficiency targets (gamified).
- Voice-enabled logging for quick start/stop.
- Bottleneck detection and heatmaps.
- Root-cause tagging for delays.

3.2. Integration Points

The Plater Time Tracking Application will integrate with the existing infrastructure at several key points:

- **GraphQL Server**: The application will communicate with the existing GraphQL server to fetch SKU information (GUIDs from the product table) and to store time tracking data. This ensures consistency with existing data structures and leverages the established API layer.
- **PostgreSQL Backend**: The GraphQL server will extend the product/user tables and add necessary endpoints to the PostgreSQL database to support the new time tracking data, SKU configurations, and efficiency targets.
- **Webapp Server**: The new application, being a standard webapp output (transpiled JS), will be deployed on the same Virtual Private Server (VPS) as the existing webapp and driverapp, ensuring efficient resource utilization and simplified deployment.

3.3. Deployment

The Plater Time Tracking Application will be deployed as a standard web application on the existing VPS that hosts the current webapp and driverapp. This approach minimizes infrastructure overhead and leverages existing deployment pipelines. The deliverables will include source code/configuration, a list of frameworks/libraries, and comprehensive documentation for setup, deployment, and data structure.

4. Technical Considerations

- **Platform**: Retool or another low-code solution is preferred for speed and maintainability.
- **Data Structure**: A preferred schema for time tracking will be specified, and existing product/user tables in PostgreSQL will be extended via GraphQL.

- **Modularity**: Connectors will be built to enable future API integrations without extensive refactoring.

5. Conclusion

The introduction of the Plater Time Tracking Application will significantly enhance VEAT's operational efficiency by automating time tracking, providing insightful analytics, and supporting basic scheduling. The proposed architecture ensures seamless integration with the current system, leveraging existing components while introducing new capabilities essential for VEAT's continued growth.

Author: Hong M. peacemanoftheworld808@gmail.com **Date**: September 23, 2025