SWOMB

Time Bought For You

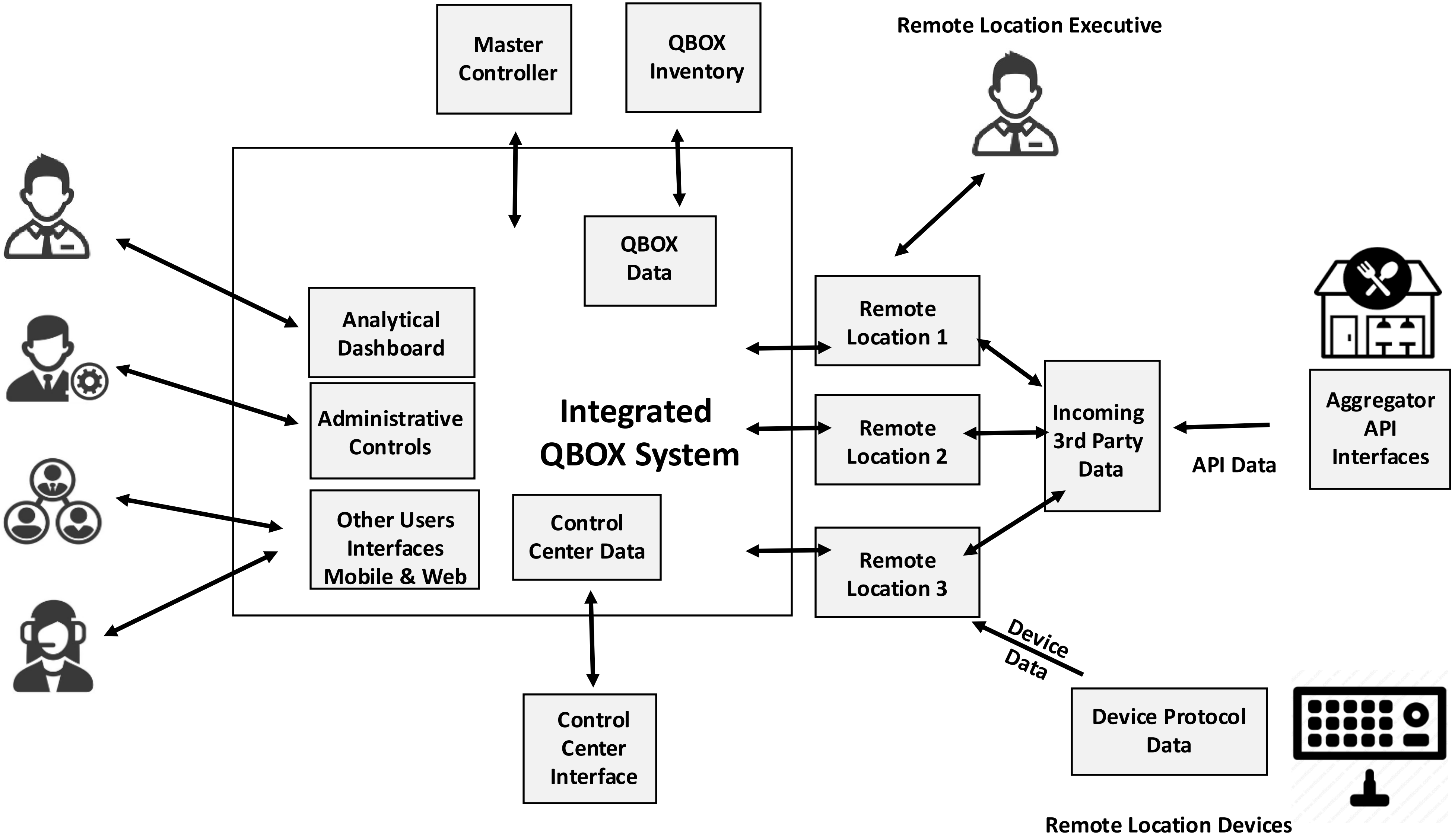
QeuBox – Architecture & High Level Design Document

Author : Vengatesan Kalimuthu
Date : 19-Aug-2025
Version : 1.1

Private & Confidential

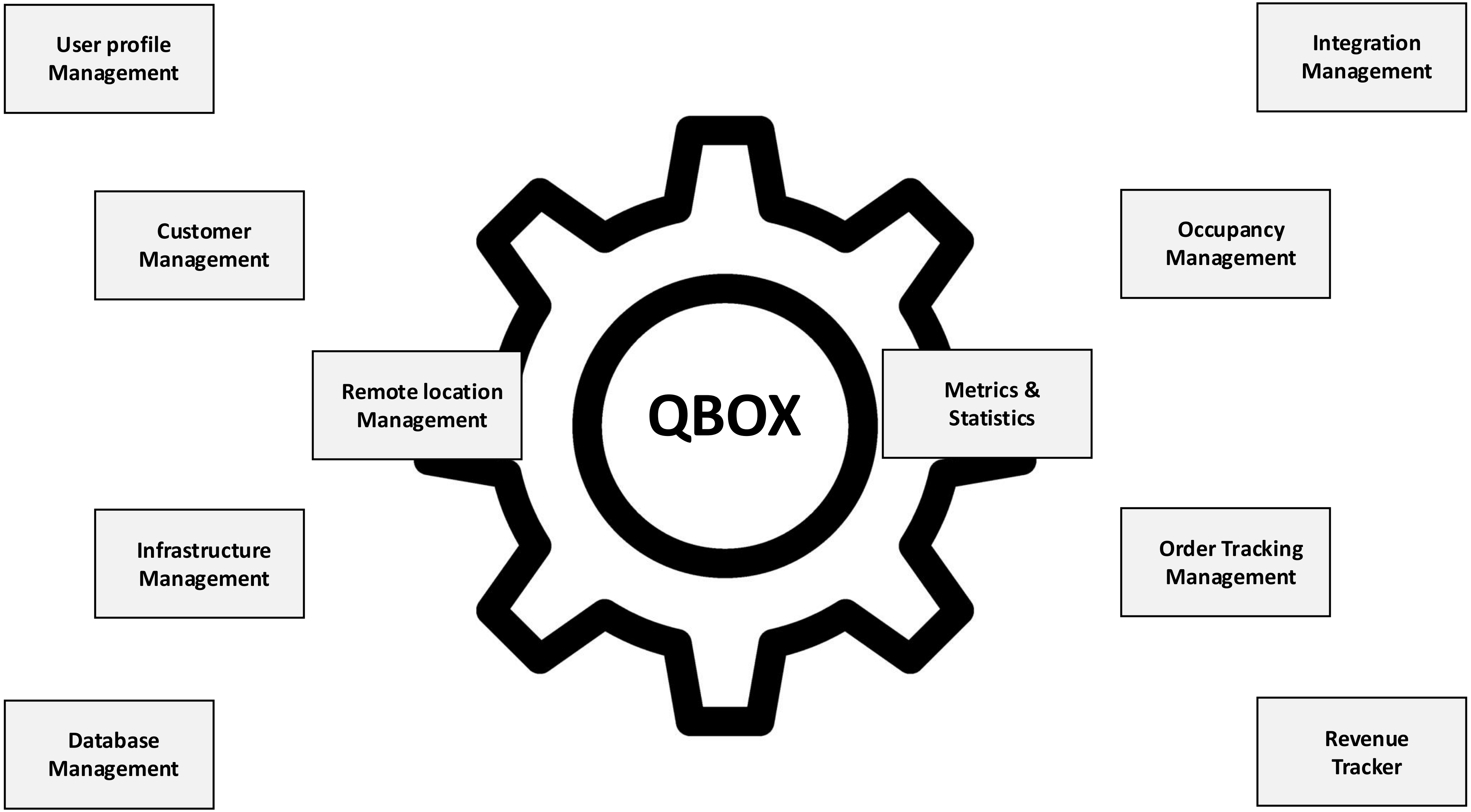
Client : ArvireTech
Chennai

Implementation Landscape

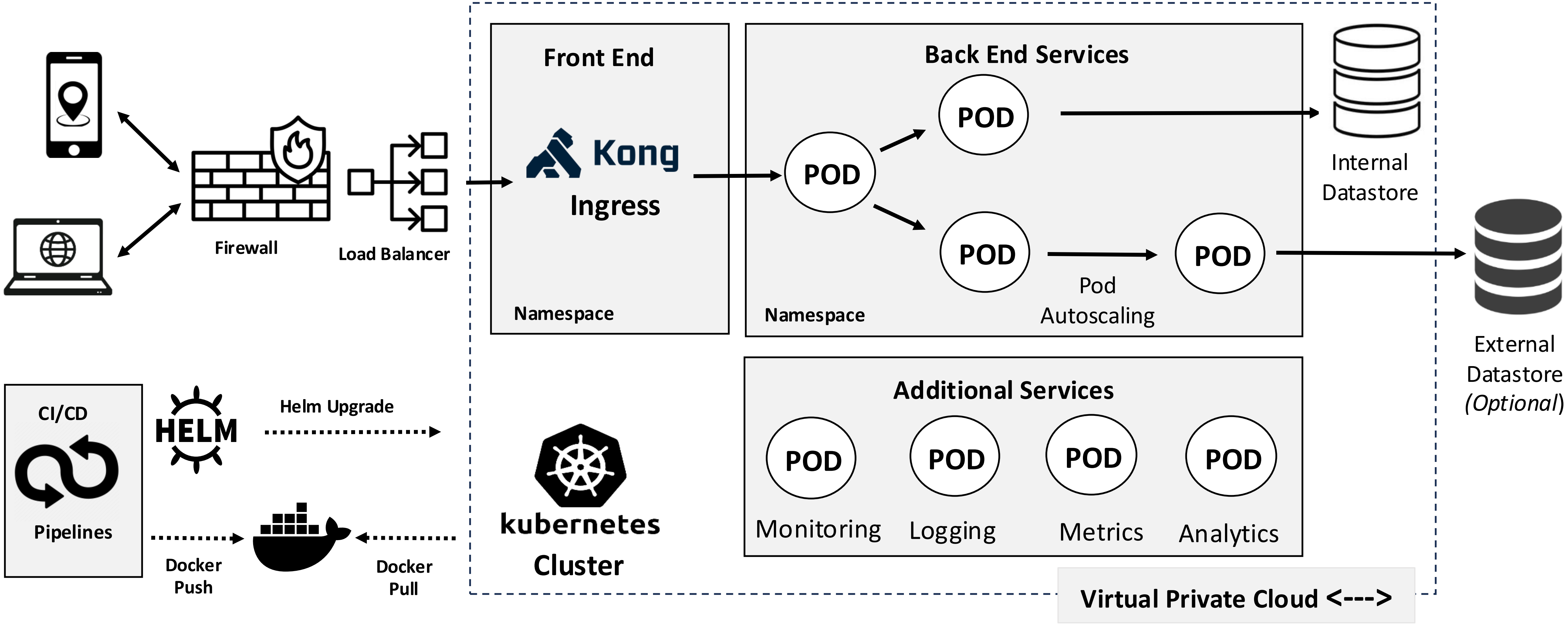


Private & Confidential

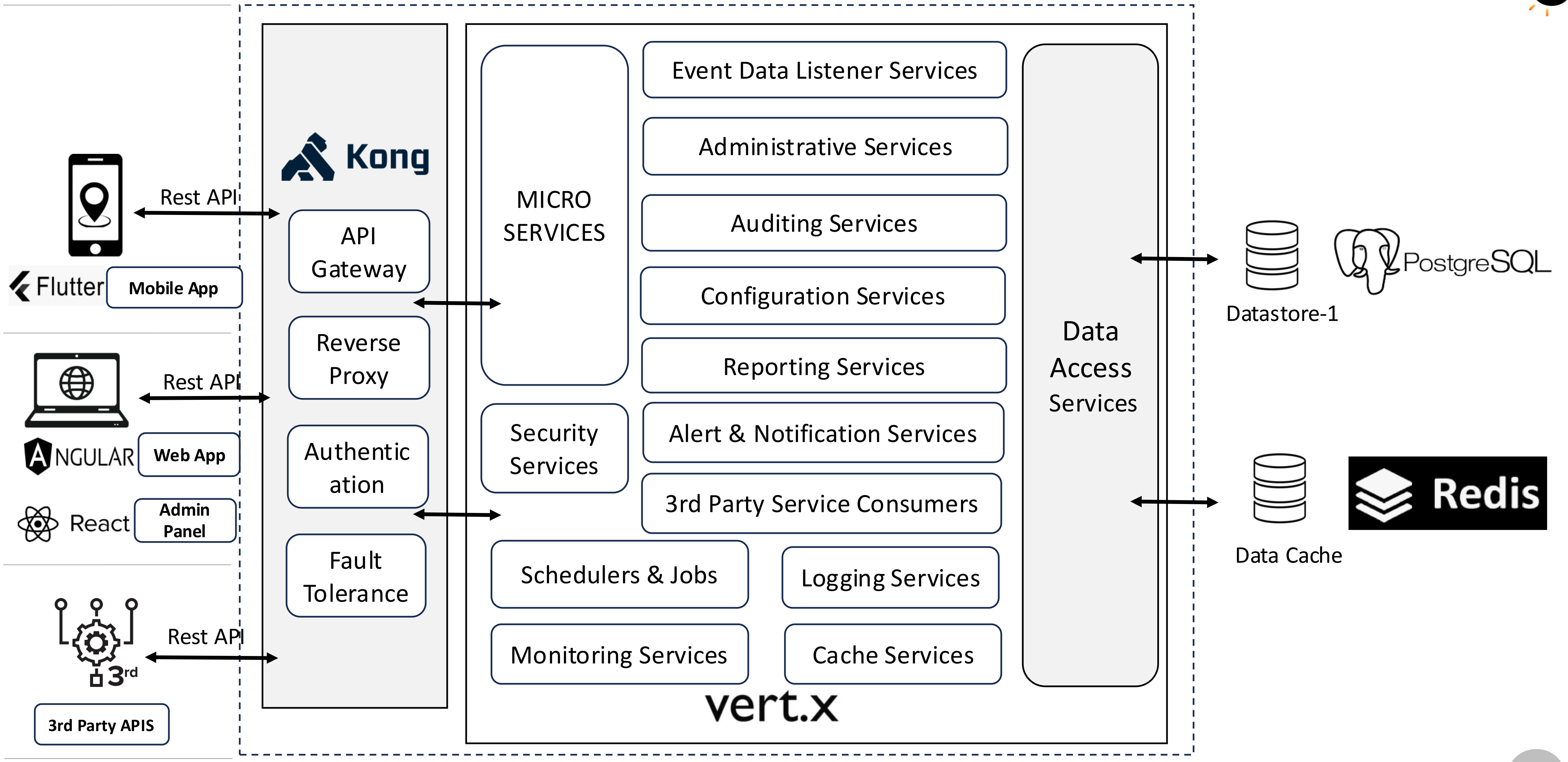
Functional Requirement



Infrastructure Architecture



Technical Architecture



Private & Confidential

Non-Functional Requirements

Performance	Ensure the applications are responsive and perform efficiently, even with a large volume of data and users.
Scalability	Design the system to handle the potential growth of the company and accommodate additional units or functionalities.
Usability	Develop an intuitive and user-friendly interface to facilitate easy adoption by employees with varying levels of technical expertise.
Data backup & recovery	Implement regular data backup and disaster recovery mechanisms to prevent data loss.
Compliance	Ensure that the applications comply with industry-specific standards and regulations pertaining to iso certifications.
Integration	Enable integration with existing and future systems or APIs
Security	Implement robust security features to protect sensitive business data and prevent unauthorized access.
Reporting and analytics	Provide comprehensive reporting and analytics tools to help management make data-driven decisions.
Hierarchical access control	Implement a hierarchical access system to control user access based on roles and responsibilities.
Documentation	Maintain detailed technical documentation for the applications to support iso compliance and facilitate future maintenance and updates.
Training and support	Provide training and support to ensure that the application users can effectively use and maintain the application.
Cloud-based	Ensure that the applications are hosted in a secure and reliable cloud environment to facilitate accessibility and scalability.
Scalable architecture	Design a modular and scalable architecture that can accommodate future expansions and changes in business requirements.
User training and onboarding	Develop comprehensive training materials and provide onboarding support to ensure all users can effectively utilize the application.

Architectural & Design Considerations

- Component based distributed architecture.
- Reactive in nature.
- Containerized builds.
- Microservice / API based.
- Clear separation of functional modules and Plug & play integrations
- The solution is being loosely coupled model, which can easily be integrated with any system, which needs LMS services.
- Role based customization
- Rule driven configurations
- Easily extendable.

Components & Technologies

- Web UI – React
- Middleware – Vert.X
- Api Gateway : Kong
- Authentication – JWT / OAuth2
- Database - Postgres
- Mobile App : Flutter
- Security - DMG with web application firewalls
- Data encryption - AES / RSA / SHA 256-bit encryption
- Hosting environment - Cloud & On prem

Microservices Detail

1) Service layer (business-facing services)

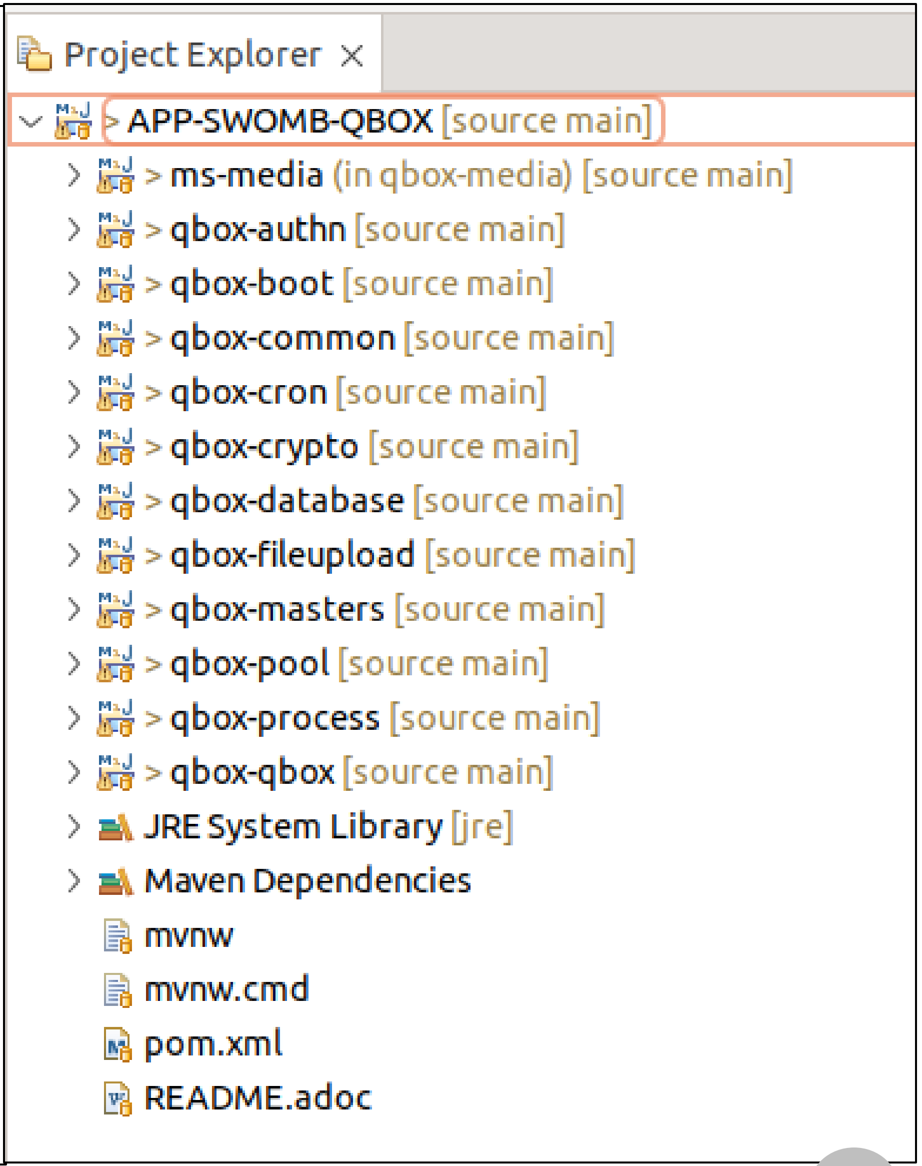
- **qbox-masters** — master/reference data service (CRUD for lookups, catalogs, statuses, etc.).
- **qbox-process** — workflow/process orchestration (domain processes, long-running ops).
- **ms-media** — media handling (ingest, transform, stream/serve media assets).
- **qbox-fileupload** — generic file ingest & storage lifecycle (upload, validate, persist).
- **qbox-pool** — pooled resources (e.g., workers/threads/tenants/queues); often supports bursty workloads.
- **qbox-authn** — authentication/identity endpoints (token issue, login flows, key rotation).

2) Platform/infra services (cross-cutting concerns)

- **qbox-crypto** — encryption, hashing, signing, key utils (used by authn/upload/media).
- **qbox-cron** — scheduled jobs (cleanup, retries, reindex, nightly rollups).
- **qbox-database** — schema migrations, seeders, DB utilities; sometimes exposes shared jOOQ/entities.

3) Shared libraries & bootstrap

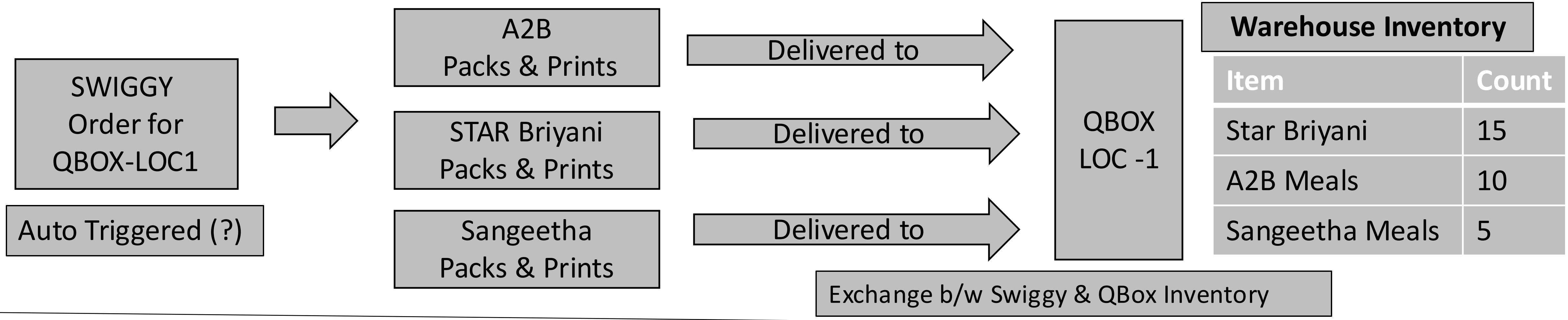
- **qbox-common** — reusable code: DTOs, errors, interceptors, utilities, constants.
- **qbox-boot** — app bootstrap & runtime wiring (parent Spring Boot starter, common config, logging).
- **qbox-qbox** — umbrella/parent aggregator module (build orchestration; sometimes the API gateway if it ships a runnable).



Project Explorer

```
APP-SWOMB-QBOX [source main]
  > ms-media (in qbox-media) [source main]
  > qbox-authn [source main]
  > qbox-boot [source main]
  > qbox-common [source main]
  > qbox-cron [source main]
  > qbox-crypto [source main]
  > qbox-database [source main]
  > qbox-fileupload [source main]
  > qbox-masters [source main]
  > qbox-pool [source main]
  > qbox-process [source main]
  > qbox-qbox [source main]
  JRE System Library [jre]
  Maven Dependencies
  mvnw
  mvnw.cmd
  pom.xml
  README.adoc
```

Use Case Flow - Delivery to QBOX



SWIGGY Order for QBOX-LOC1

Auto Triggered (?)

A2B Packs & Prints

STAR Briyani Packs & Prints

Sangeetha Packs & Prints

Delivered to

Delivered to

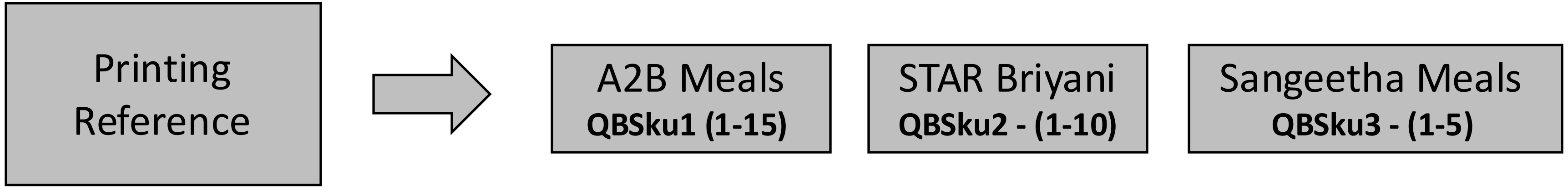
Delivered to

QBOX LOC -1

Warehouse Inventory

Item	Count
Star Briyani	15
A2B Meals	10
Sangeetha Meals	5

Exchange b/w Swiggy & QBox Inventory

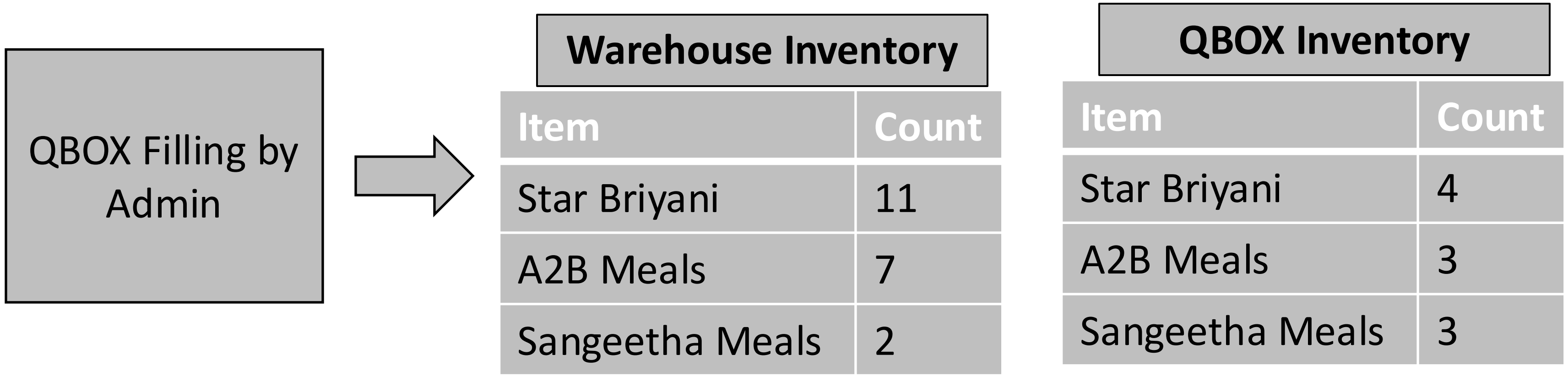


Printing Reference

A2B Meals
QBSku1 (1-15)

STAR Briyani
QBSku2 - (1-10)

Sangeetha Meals
QBSku3 - (1-5)



QBOX Filling by Admin

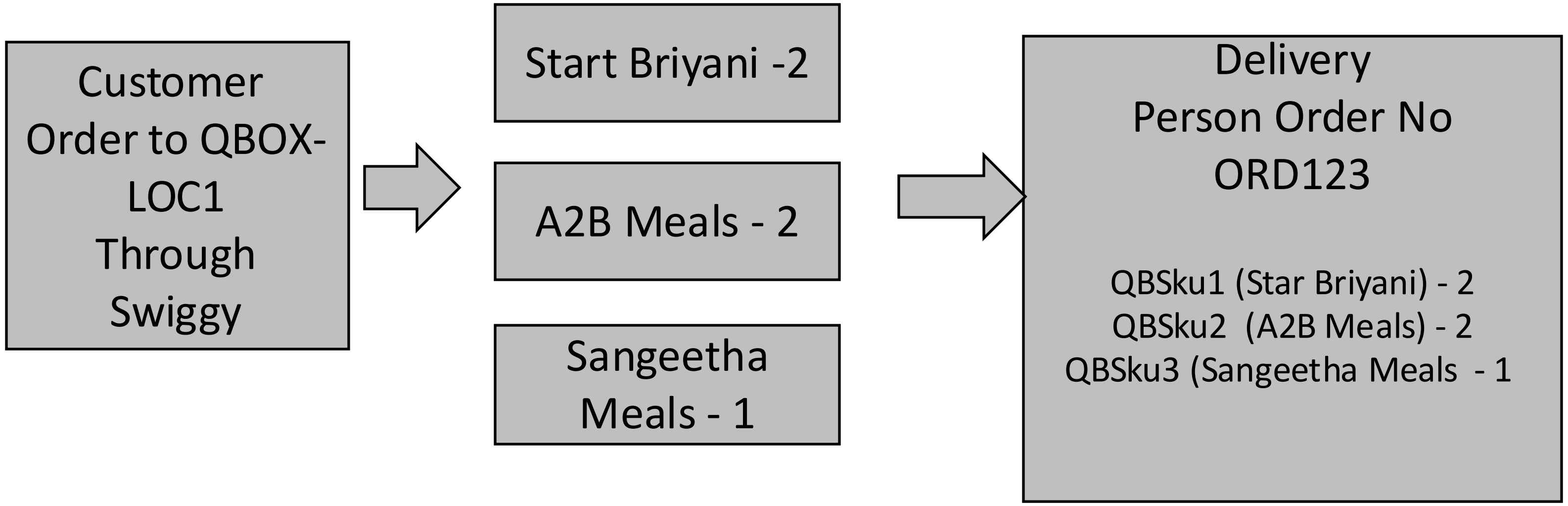
Warehouse Inventory

Item	Count
Star Briyani	11
A2B Meals	7
Sangeetha Meals	2

QBOX Inventory

Item	Count
Star Briyani	4
A2B Meals	3
Sangeetha Meals	3

Use Case Flow - Delivery to QBOX


Customer Order to QBOX-LOC1 Through Swiggy

Start Briyani -2

A2B Meals - 2

Sangeetha Meals - 1

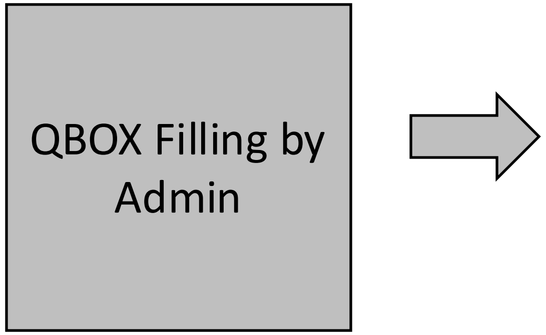
Delivery Person Order No ORD123

QBSku1 (Star Briyani) - 2
QBSku2 (A2B Meals) - 2
QBSku3 (Sangeetha Meals - 1

QBOX Inventory

Item	Pre Scan	Post Scan
Star Briyani	4	2
A2B Meals	3	1
Sangeetha Meals	3	2

Dashboard for Current Window


QBOX Filling by Admin

Warehouse Inventory

Item	Count
Star Briyani	9
A2B Meals	4
Sangeetha Meals	2

QBOX Inventory

Item	Count
Star Briyani	4
A2B Meals	4
Sangeetha Meals	2

Delivered Qty

Item	Count
Star Briyani	2
A2B Meals	2
Sangeetha Meals	1