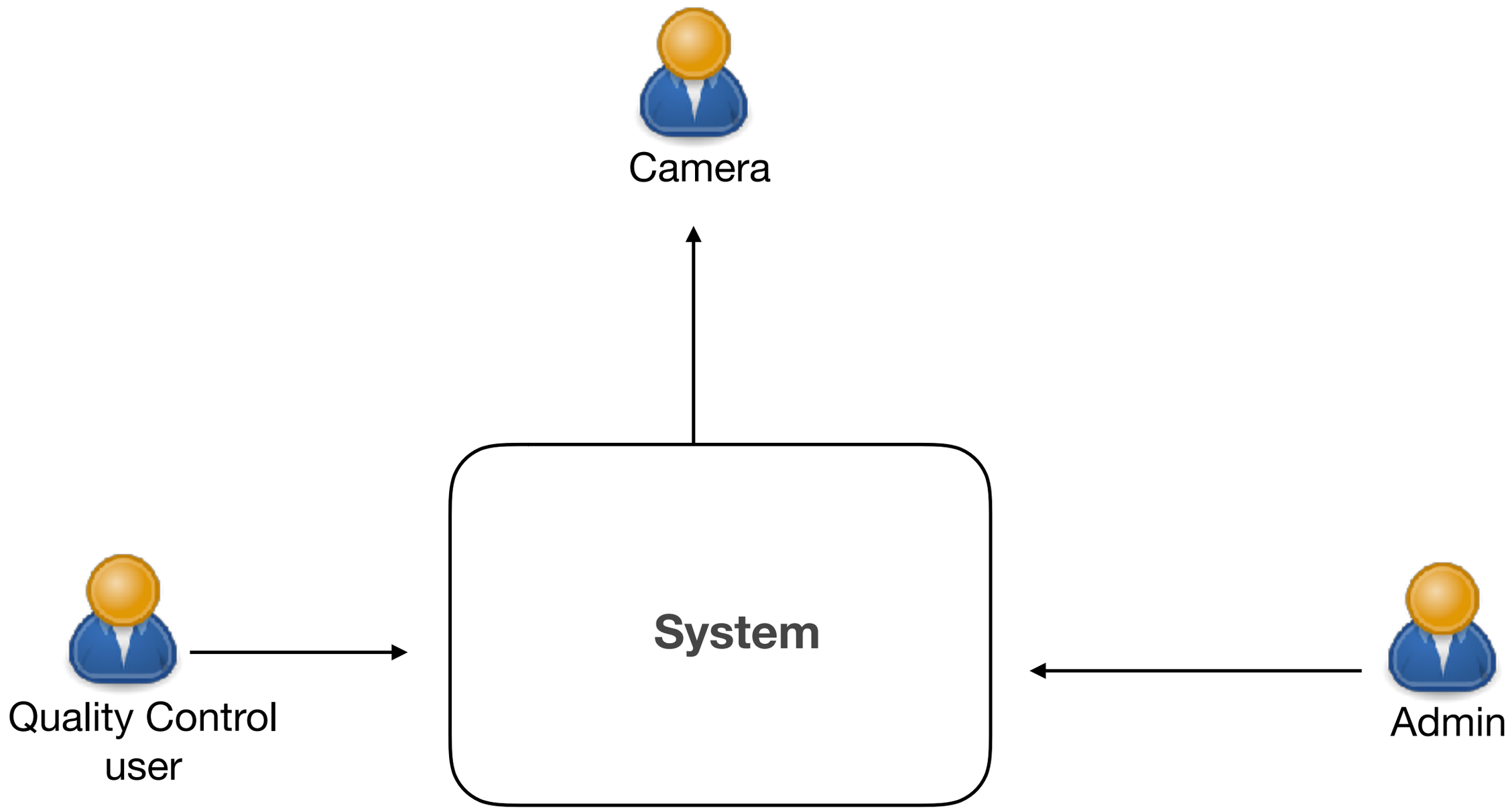
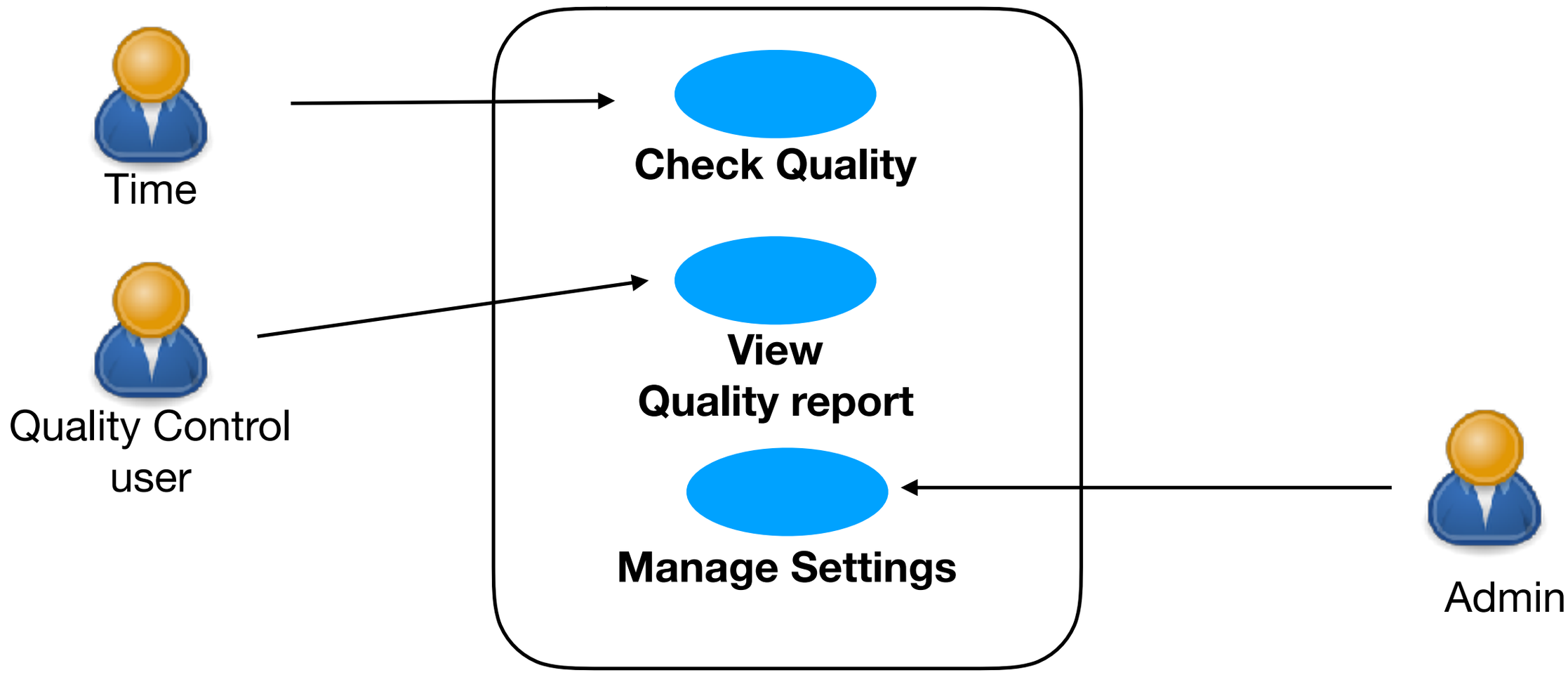
Case study

Context View



Functional view



Quality View

Quality	Source (who)	Stimulus (action)	Artifact (which)	Environment (context)	Response (output)	Measure (scale)
Performanc e	As a user	I want to Check Quality of a Circuit	Using the Defect Detection System	During peak load.	The Quality of Circuit Board is verified	In < 3 secs
Robustness	System Admin	enters incorrect settings	In the portal	During Normal load.	the system prints an error message and execution stops.	User is able to correct errors at rate of 1 minute/error
Maintainabi lity	Developer	add additional pre processing logic	In the Image processing	During maintenance	The pre processing logic is added	In < 2 man days
Security	unknown identity	requests to add a new settings	In the portal	During Normal load.	block access to the data and record the access attempts	100% probability of detecting the attack, 100% probability of denying access
Availability	The Camera	Failed	In the Data Collection Device	During Operational Hours	Secondary is made Primary	In < 2 minutes

Constraints view

The Portal should work on IE 11, Chrome and Firefox

The System support windows platforms

The architecture should support Microsoft SQL as database servers.

The architecture should be able to work behind firewalls.

AuthN and AuthZ will be handled by Kuali Identity Management (KIM)

Assumptions View

Will use open source Technologies only

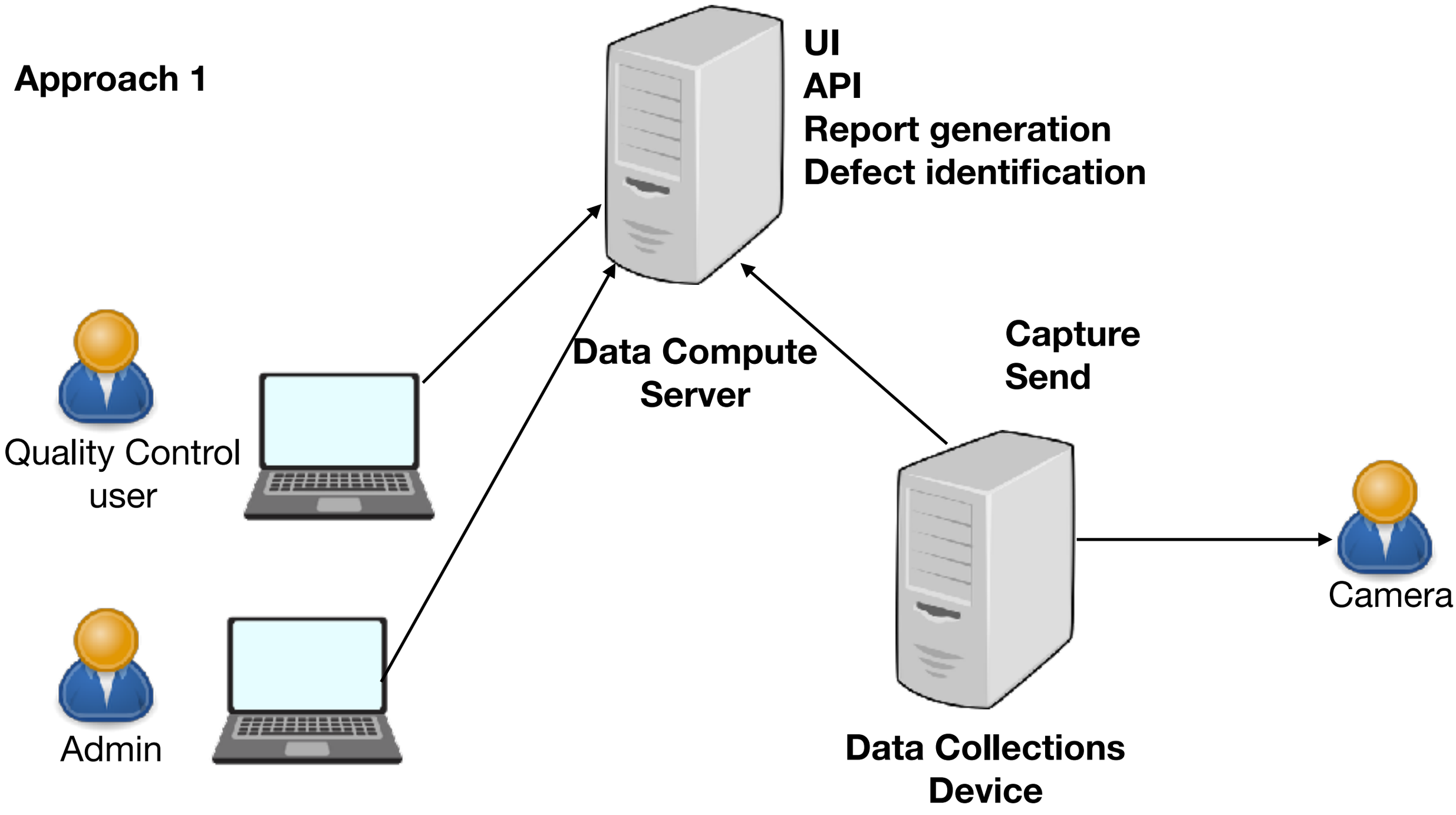
This will be a Java application and will run in an Apache Tomcat container

---

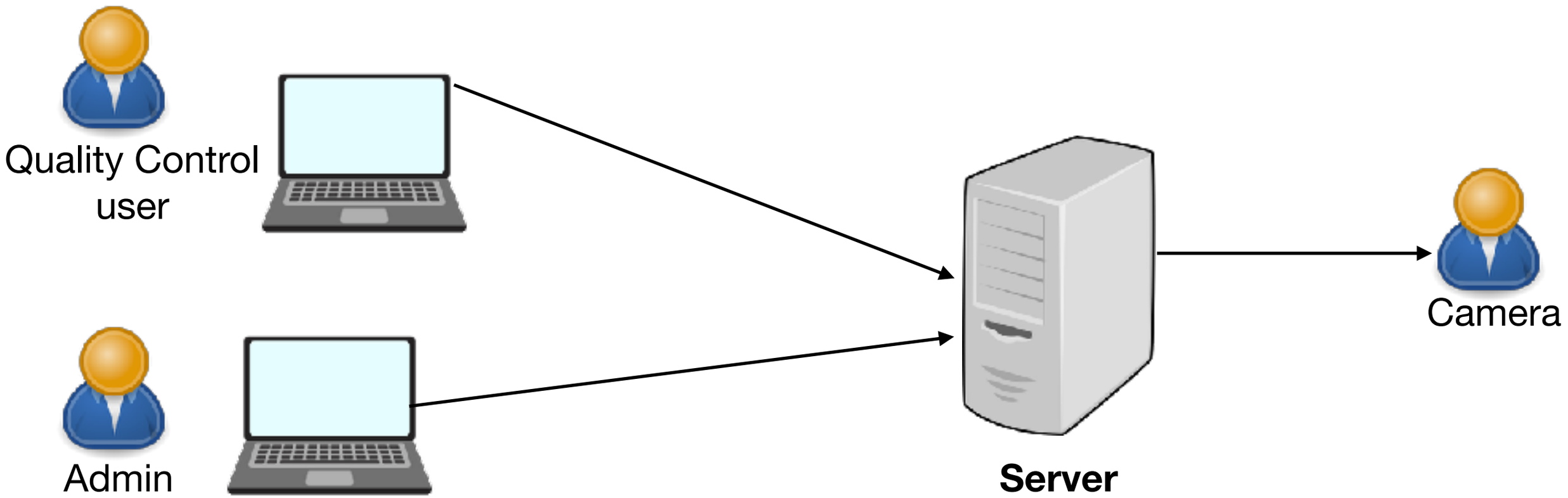
Logical View

System Decomposition

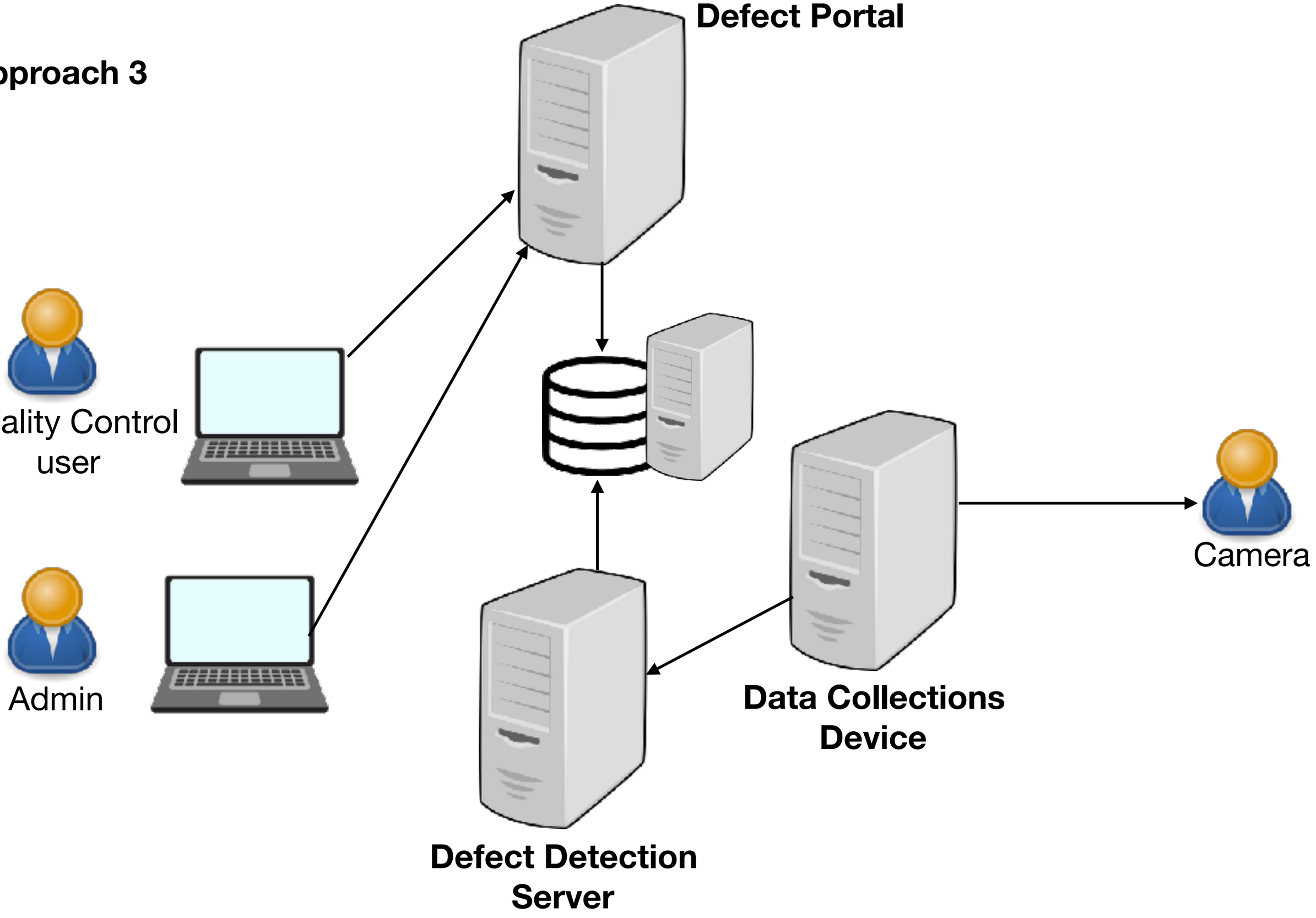
Approach 1



Approach 2



Approach 3

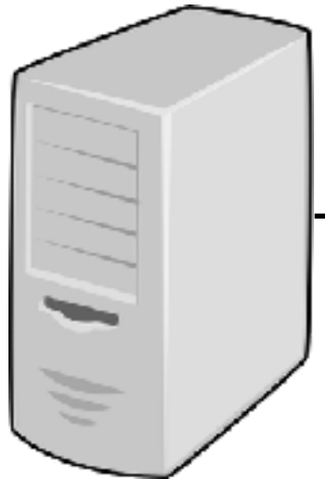

Defect Portal
Quality Control user
Admin
Camera
Data Collections Device
Defect Detection Server

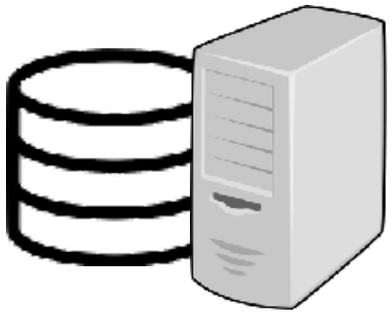
Approach 4

UI
API
Report generation

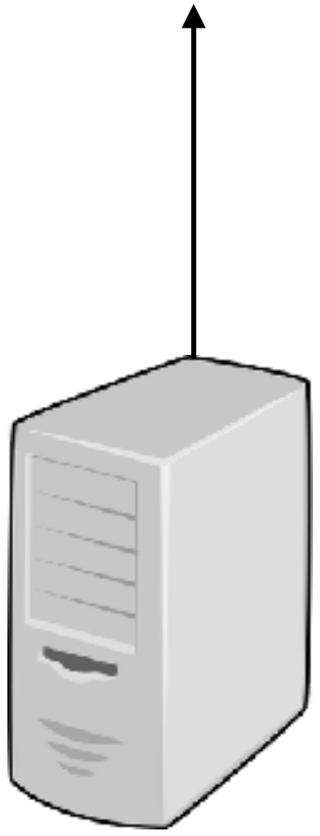
Web Portal

Quality Control user

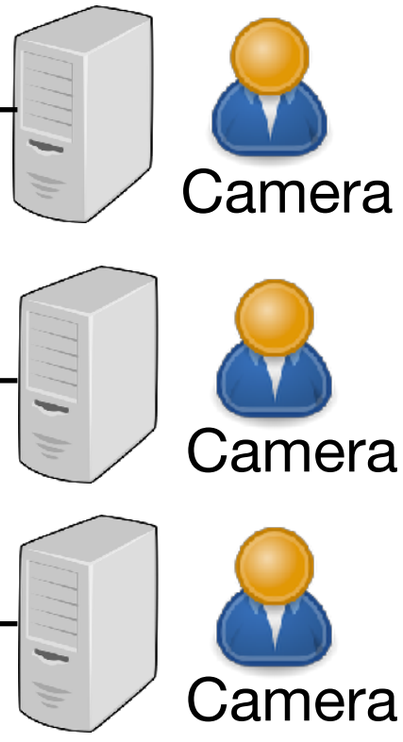




Admin



Defect Detection Server



Data Collections Device

Approach 4

UI Portal

API App

Report Generation App

Quality Control user

Admin

http: Rest

Getconfig

?

?

Quality Check results

?

Camera

Camera

Camera

Defect Detection App

Data Collection App

Data Collection Service

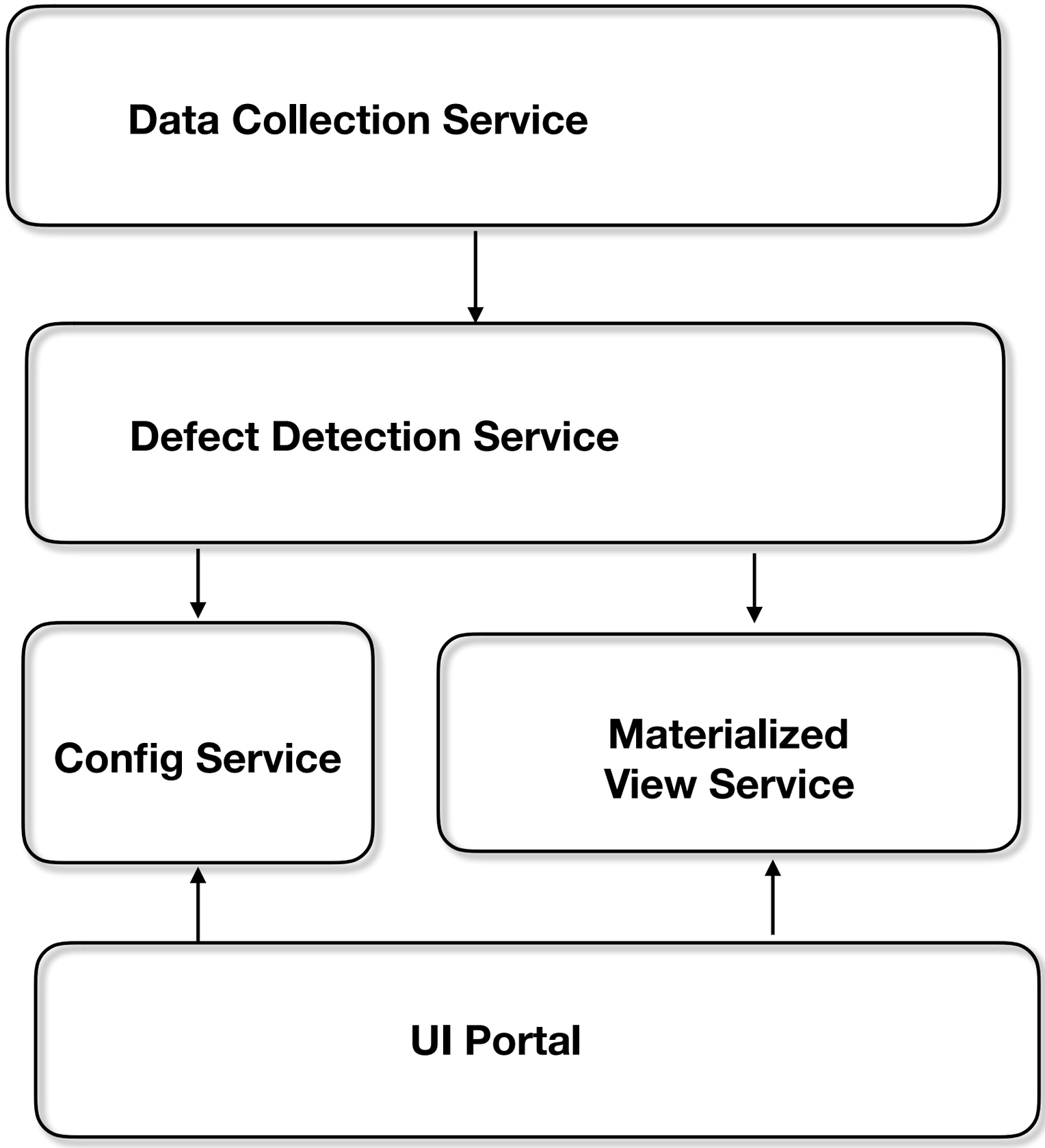
Defect Detection Service

Config Service

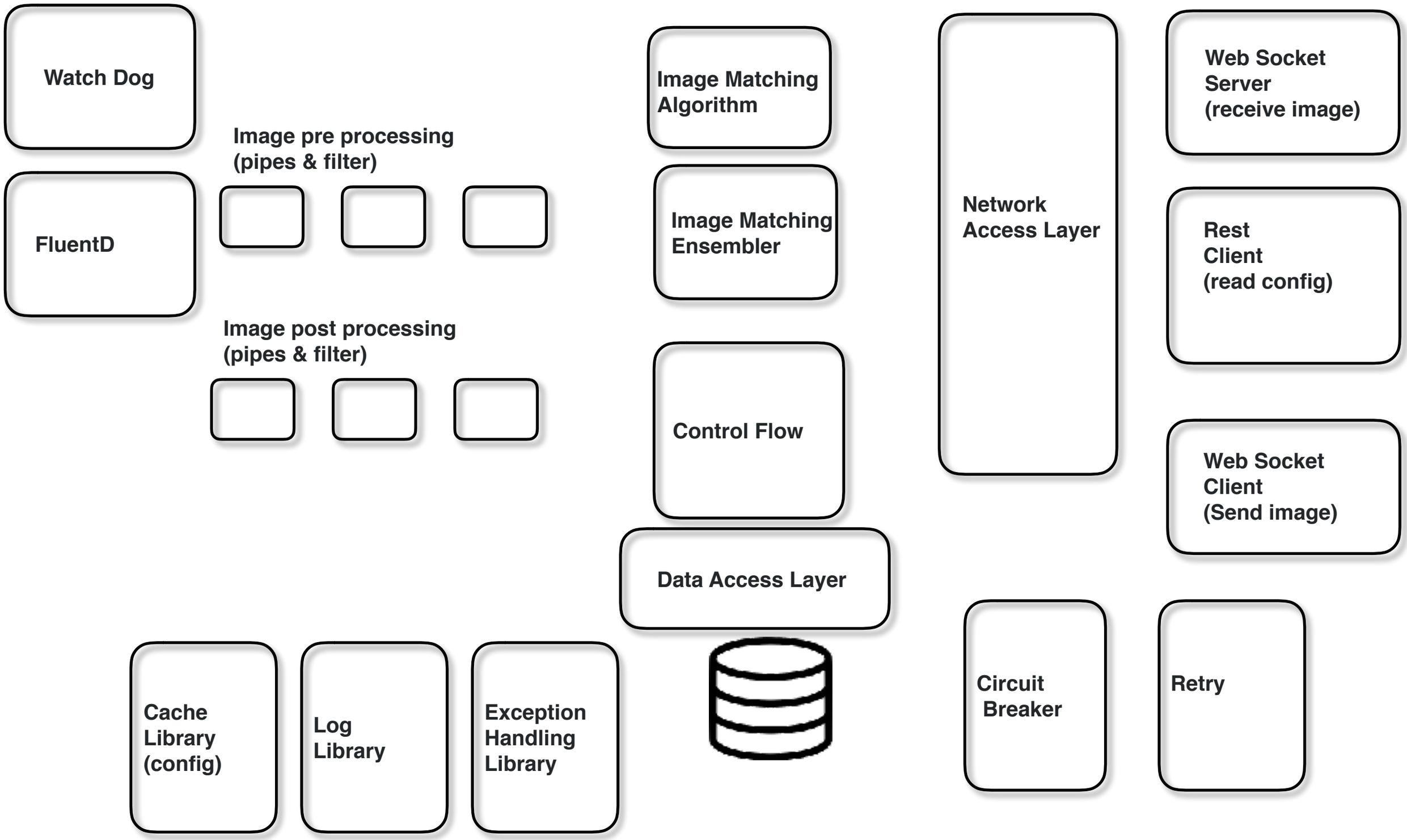
Materialized
View Service

UI Portal

Decide on coupling (dependancy)


Data Collection Service
Defect Detection Service
Config Service
Materialized
View Service
UI Portal

Application Decomposition



Data Collection Service

Defect Detection Service

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

Config Service

Materialized View Service

UI Portal

Data Collection Service

<<write>>

Defect Detection Service

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

<<read>>

<<write>>

Distributed Queue

Config Service

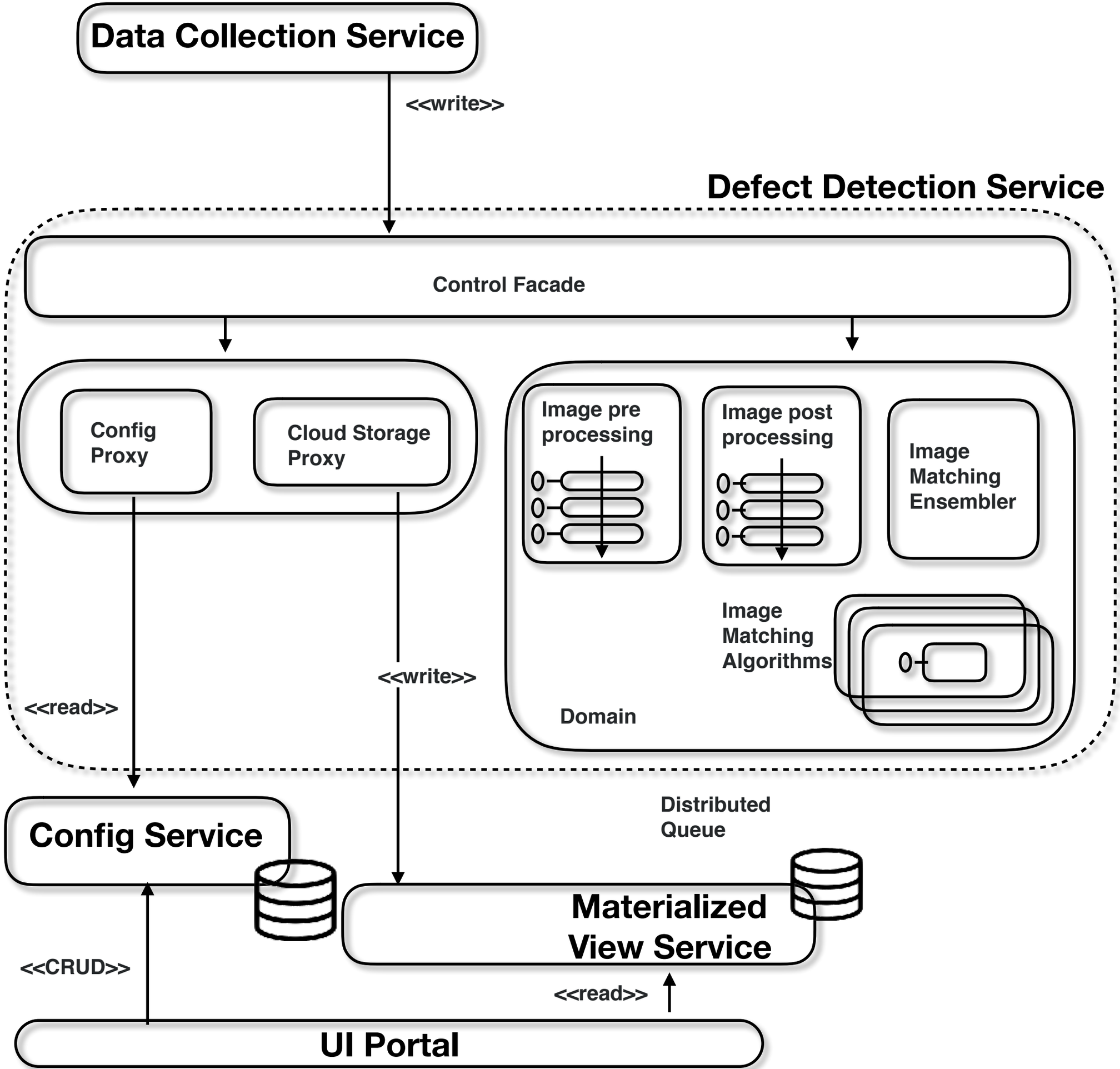
Materialized View Service

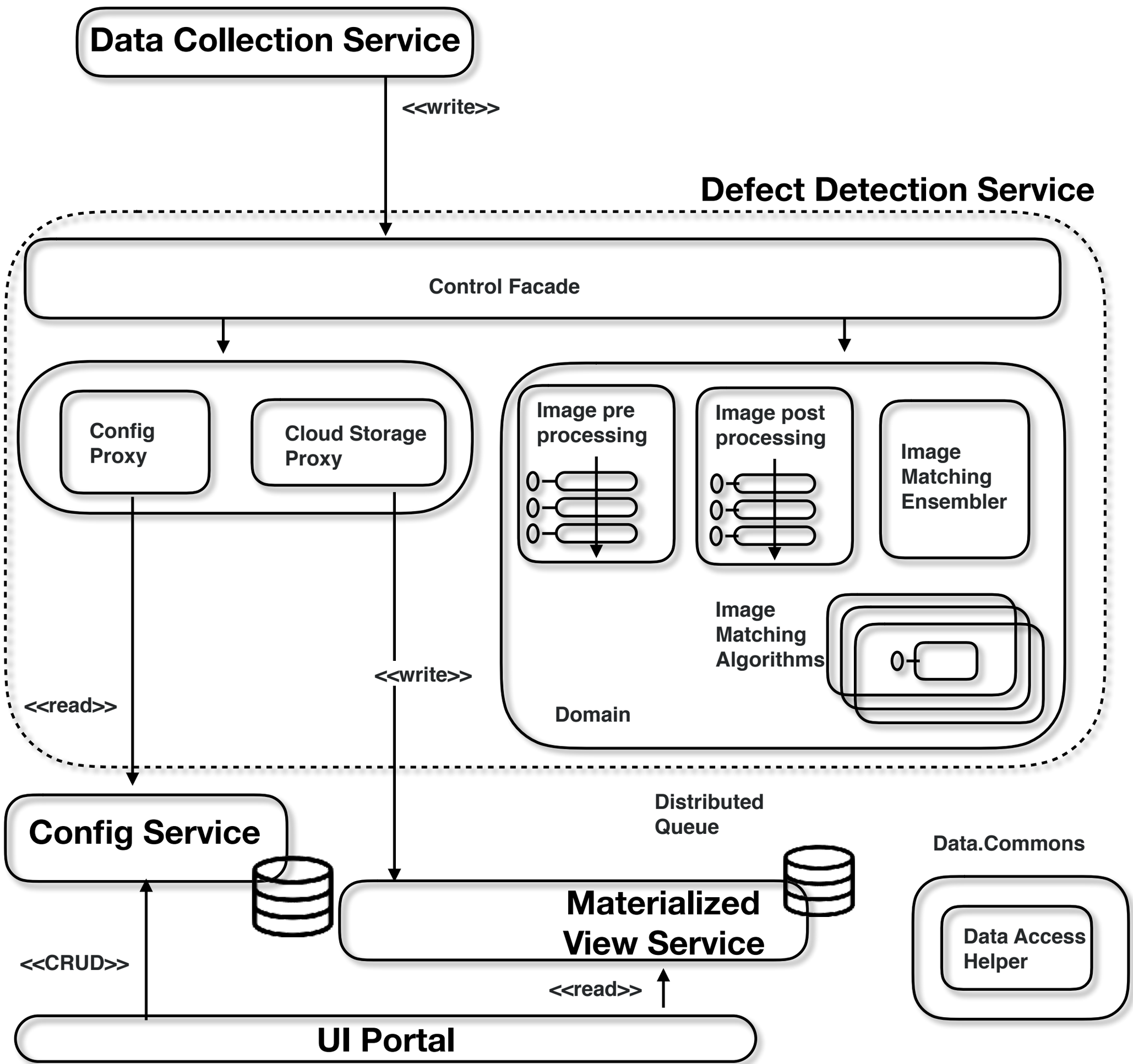
<<CRUD>>

<<read>>

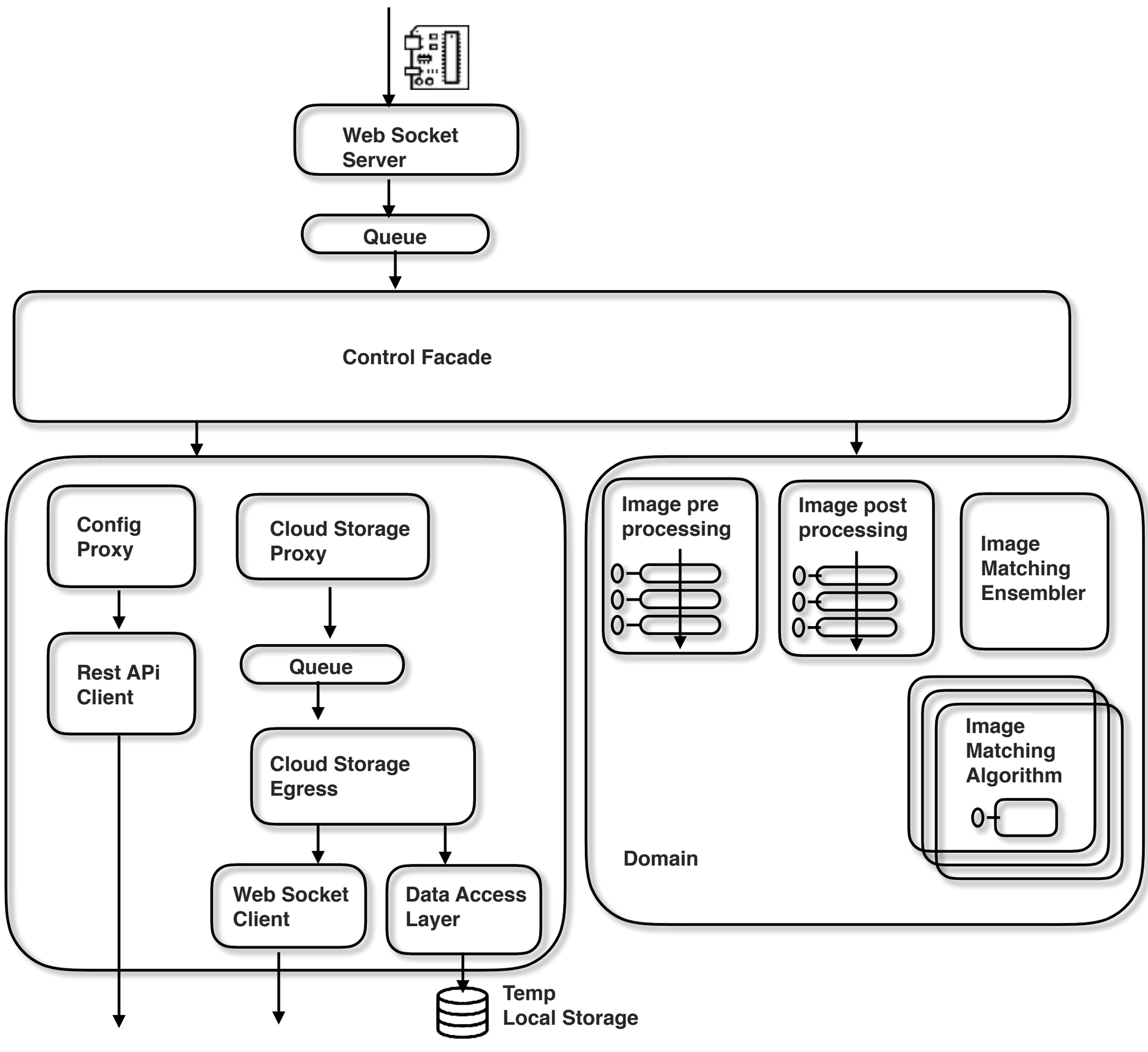
UI Portal

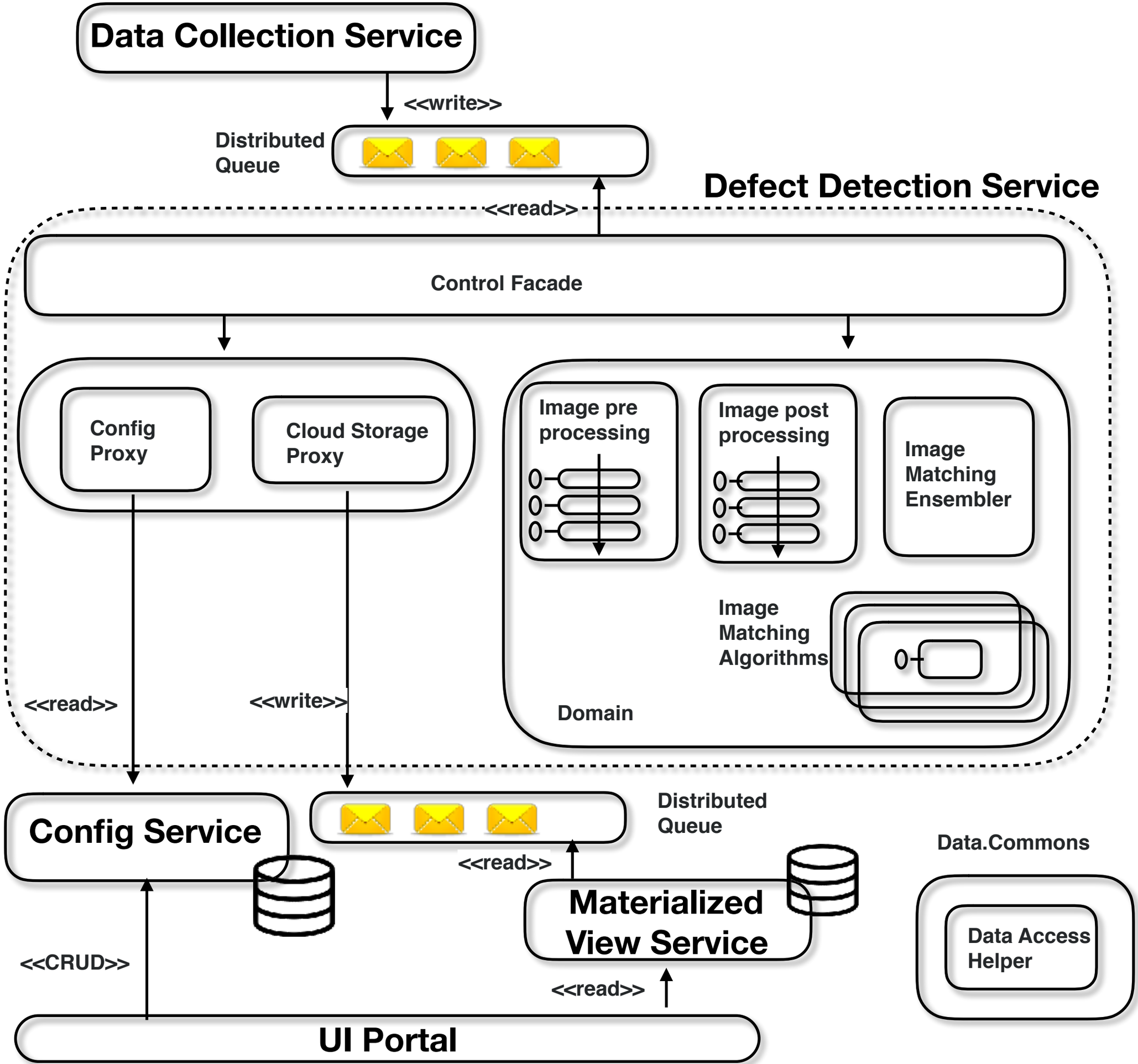
Choose Persistence


Data Collection Service
<<write>>
Defect Detection Service
Control Facade
Config Proxy
Cloud Storage Proxy
Image pre processing
Image post processing
Image Matching Ensembler
Image Matching Algorithms
Domain
<<read>>
<<write>>
Config Service
Distributed Queue
Materialized View Service
<<CRUD>>
<<read>>
UI Portal


Data Collection Service
<<write>>
Defect Detection Service
Control Facade
Config Proxy
Cloud Storage Proxy
Image pre processing
Image post processing
Image Matching Ensembler
Image Matching Algorithms
Domain
<<read>>
<<write>>
Config Service
Distributed Queue
Materialized View Service
Data.Commons
Data Access Helper
<<CRUD>>
<<read>>
UI Portal

Choose Communication


Web Socket Server
Queue
Control Facade
Config Proxy
Rest APi Client
Cloud Storage Proxy
Queue
Cloud Storage Egress
Web Socket Client
Data Access Layer
Temp Local Storage
Image pre processing
Image post processing
Image Matching Ensembler
Image Matching Algorithm
Domain


Data Collection Service
<<write>>
Distributed Queue
Defect Detection Service
<<read>>
Control Facade
Config Proxy
Cloud Storage Proxy
Image pre processing
Image post processing
Image Matching Ensembler
Image Matching Algorithms
Domain
<<read>>
<<write>>
Config Service
Distributed Queue
<<read>>
Data.Commons
Materialized View Service
Data Access Helper
<<CRUD>>
<<read>>
UI Portal

Data Collection Service

<<write>>

Distributed Queue

Defect Detection Service

<<read>>

Network.Commons

Queue Receiver <<Adapter>>

Queue Publisher <<Adapter>>

RestClient <<Adapter>>

Circuit Breaker

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

<<read>>

<<write>>

Config Service

Distributed Queue

<<read>>

Data.Commons

Materialized View Service

Data Access Helper

<<CRUD>>

<<read>>

UI Portal

Address Cross Cutting Concerns

Network.Commons

Queue Receiver <<Adapter>>

Queue Publisher <<Adapter>>

RestClient <<Adapter>>

Circuit Breaker

CrossCutting.Commons

Log Library

Cache Library (config)

Exception Handling Library

Data Collection Service

<<write>>

Distributed Queue

Defect Detection Service

<<read>>

Control Facade

Config Proxy

Cloud Storage Proxy

Image pre processing

Image post processing

Image Matching Ensembler

Image Matching Algorithms

Domain

<<read>>

<<write>>

Config Service

Distributed Queue

<<read>>

Materialized View Service

<<CRUD>>

<<read>>

UI Portal

Data.Commons

Data Access Helper

Choose Technology Stack

Data Collection Service

<<write>>

Distributed Queue

RabbitMQ

<<read>>

Network.Commons

python

Defect Detection Service

python

<<read>>

<<write>>

Data.Commons

python

Config Service

python

RabbitMQ

<<read>>

Materialized View Service

python

CrossCutting.Commons

python

<<CRUD>>

<<read>>

UI Portal

python

Flask

Apache Web Server

Physical View
(Infrastructure)

Data Collection Service

Distributed
Queue Server

Defect Detection Service

Materialized
View Service

Config Service

UI Portal

APi Gateway

Cloud lb

Cluster

Kubernetes ingress/ **Istio Ingress** / nginx reverse proxy

vnet

Data Collection Service

UI Portal

Distributed Queue Service

Defect Detection Service

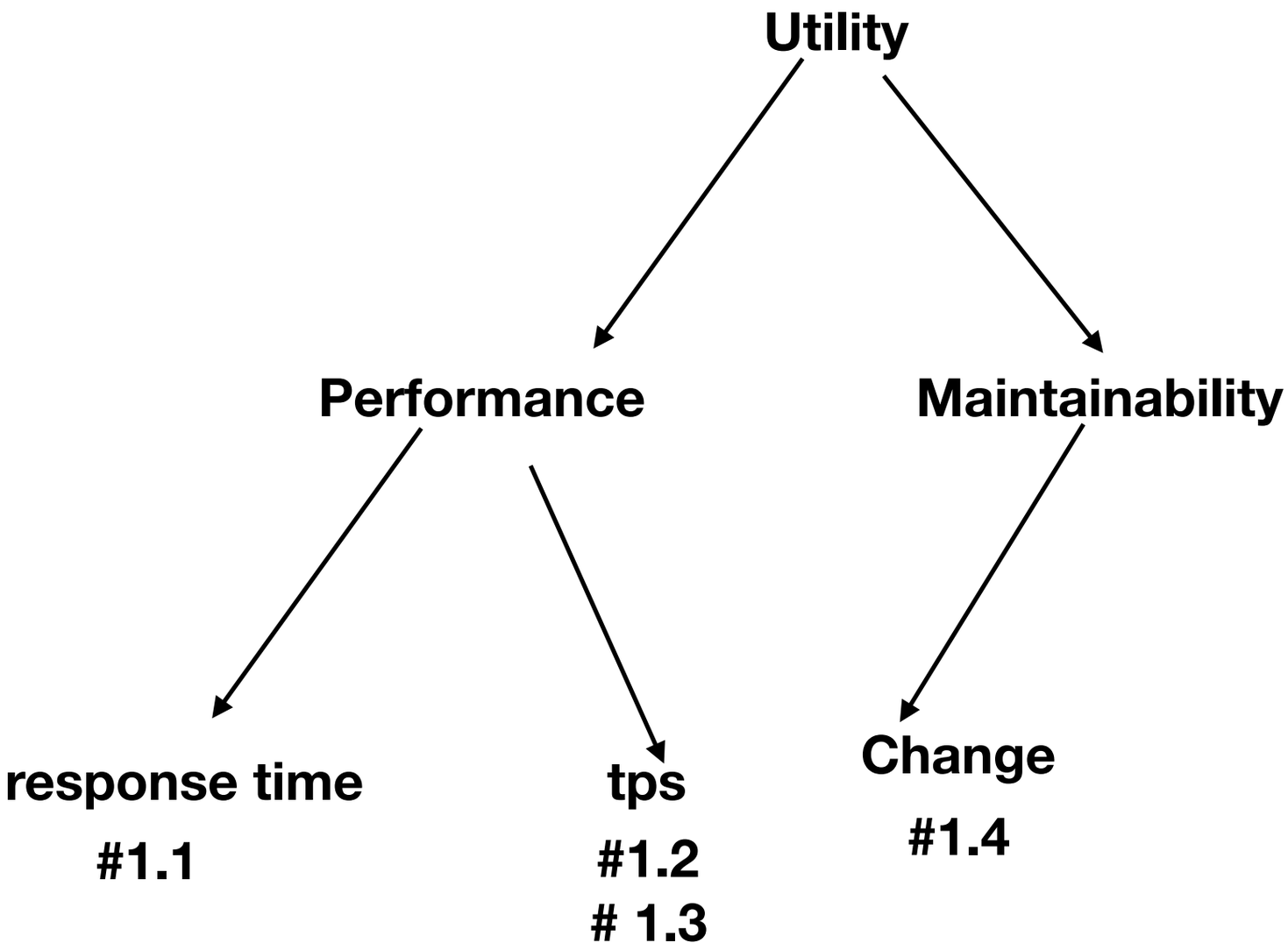
Config Service

Database Service

Materialized View Service

Architecture Justification

1. Used pipes and filter in image pre and post processing
2. Used Message Queue to scale input
3. Used cache to hold config data
4. Used adapters to remove coupling with vendor libraries
5. Debug and exceptions log
6. Circuit breaker for fail fast



1.1 The data collection device should be able capture images of CB at 2 images/sec
1.2 The system should be able to capture images from at least 10 belts during peak load
1.3 the defect detection system should be able to match CB for 100 images during peak load
1.4 Developer should be able to add additional pre processing logic to image processing in less than 3 man days.

1. Used pipes and filter in image pre and post processing
2. Used Message Queue to scale input
3. Used cache to hold config data
4. Used adapters to remove coupling with vendor libraries
5. Debug and exceptions log
6. Circuit breaker for fail fast

1.1 The data collection device should be able capture images of CB at 2 images/sec
1.2 The system should be able to capture images from at least 10 belts during peak load
1.3 the defect detection system should be able to match CB for 100 images during peak load
1.4 Developer should be able to add additional pre processing logic to image processing in less than 3 man days.

SC#	A#	Trade off	Risks
1.1	A2		
1.2	A2		
1.3			May need GPU
1.4	A1		