Date	23 September 2022
Team ID	PNT2022TMID12210
Project Name	Project – Crude Oil Price Prediction
Maximum Marks	2 Marks

Solution Architecture Definiton

for

Crude Oil Price Prediction

Version 1.0 approved

TABLE OF CONTENTS

1 INTRODUCTION

The agreement between the client and the developer regarding the software solution architecture product is put forth.

1.1 PURPOSE

This document describes the technical architecture of the solution through different views to expose the concepts, constraints and mechanics behind it. It does not set out a list of responsibilities of the parties or the deliverables of the product.

1.2 DISCLAIMERS

The architecture specifications captured in this document are not fixed and are likely to change during the project phases.

Each version contains the architecture views and decisions for that moment in time, but do not necessarily reflect the same architecture components and integrations that will be delivered for each release.

This is mainly due to the fact that the view of the system evolves during its development. This is especially true during the Foundation project phase, as only a high-level view and understanding of the solution is available.

The architecture view, key design decisions and integrations are most likely to change during the Engineering phase.

1.3 OVERVIEW

The Solution Architecture Definition document describes the subsystems and components of the solution by presenting a number of architectural views. Each view shows a different aspect of the system to address different concerns. These are as follows:

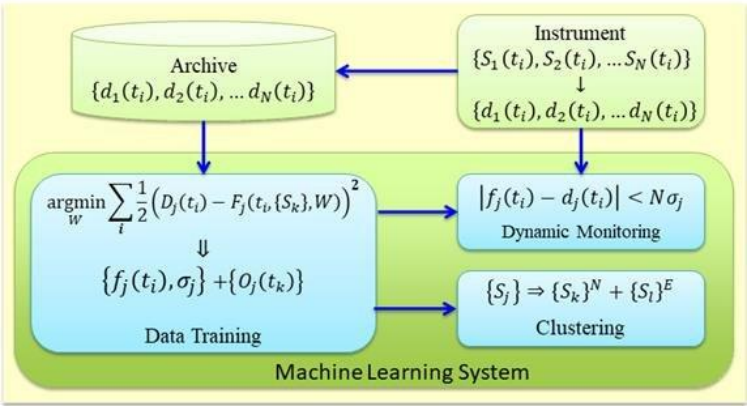
- **Assumptions and Constraints** – describes any limitations and factors that must be taken into account when designing the system.
- **Context View** – shows at the highest level how the proposed system interacts with external systems and user groups.
- **Project View** – maps key functionality to releases and milestones.
- **Functional View** – describes the top-level functionality that the system should implement and summarises the architecturally significant Use Cases (i.e. those which

influenced the overall architectural design of the system) along with how the key Use Cases will be realized.

- **Process View** – describes the top-level workflows and processes to be followed within the system.
- **Non-functional View** – describes specific system changes that allow meeting the non-functional requirements, which are defined outside of the SAD.
- **Logical View** – defies the functional decomposition of the system into smaller subsystems
- **Interface View** – defines the system interfaces required in the solution.
- **Design View** – this section describes any lower level design detail required to support the implementation of the system.
- **Physical View** – shows the hardware infrastructure on which the software runs.
- **Deployment View** – shows the mapping of the logical components onto the physical infrastructure.
- **Operational View** – describes the elements of the solution that will support the operations and support teams when the system goes into production.
- **Security View** – describes elements and functionality that impact the overall security of the application.
- **Data View** – shows the main data structures and how the entities relate to each other

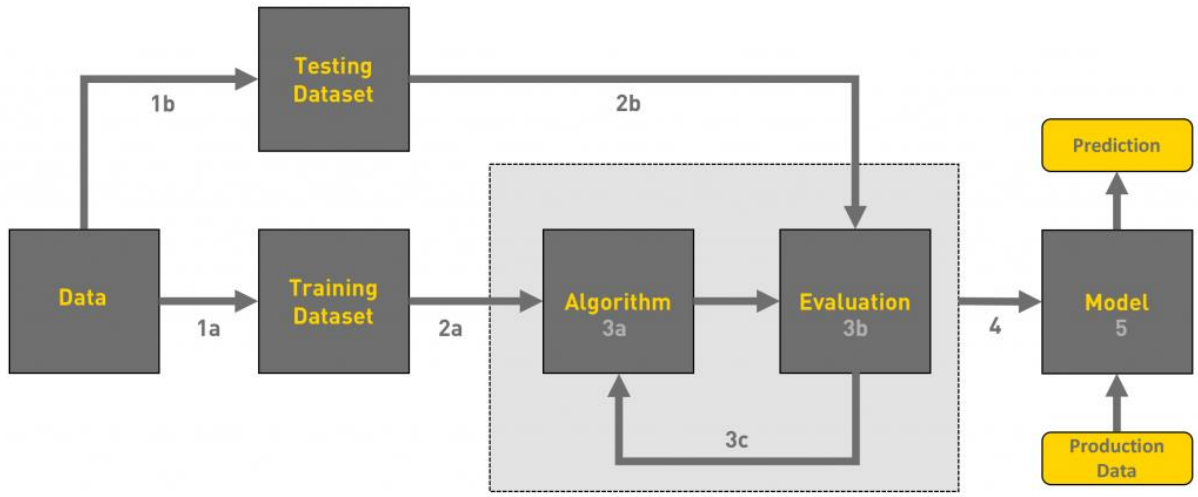
2 CONTEXT VIEW

This view gives a high level representation of the system, the different user types and interactions with external entities. It describes the boundaries of the solution.



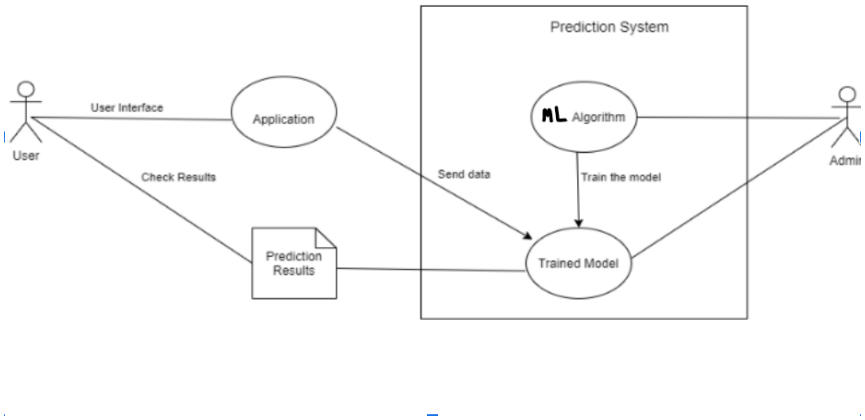
3 PROJECT VIEW

This section shows how key functionality relevant to the solution architecture maps to releases and milestones.



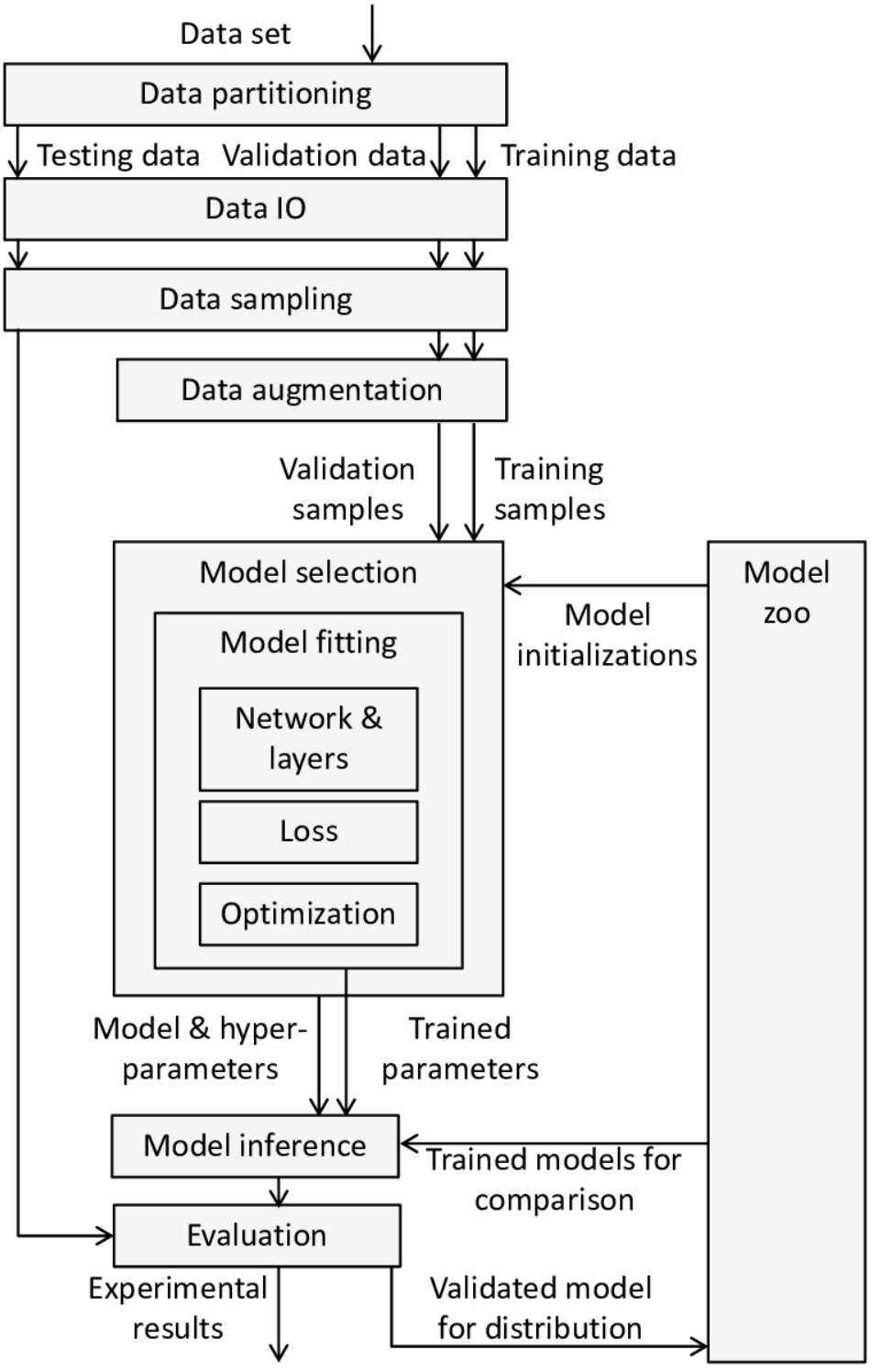
4 FUNCTIONAL VIEW

This section describes the key functional areas of the project. The goal is to provide context around the architecture – all software performs some functionality and the definition of this functional scope is a very important factor to define the architecture.



5 PROCESS VIEW

The intent of the process view is to show how the various processing steps within the system fit together to implement the overall functional requirements. This is necessary if the system relies on workflow processes, forked or parallel processing mechanisms. The following processes are significant:



6 NON FUNCTIONAL VIEW

This section describes architecturally significant changes that enable the solution to achieve the agreed non-functional requirements (NFRs). Each change is mapped to the corresponding NFR category, which is based on the ISO/IEC 25010-2011 product quality model.

NFRs are documented and maintained in the Non Functional Requirements Definition and will not be repeated here. In case of duplication, the Non Functional Requirements Definition takes precedence.

Performance Easy tracking of records and updating can be done. All the requirements relating to performance characteristics of the system are specified in the section below. There are two types of requirements.

1.Static Requirements:

These requirements do not impose any constraints on the execution characteristics of the system. They are:

A) Number of Terminals: The software makes use of an underlying database that will reside at the same system, while the front end will be available to the administrative computer.

B) Number of Users: The number of users can be administrator only, but this software can be extended to applications for almost all staff members of the organization.

2. Dynamic Requirements:

These specify constraints on the execution characteristics of the system. They typically include response time and throughout of the system. Since these factors are not applicable to the proposed software, it will suffice if the response tine is high and the transactions are carried out precisely and quickly. Reliability: The software will not be able to connect to the database in the event of the server being down due to a hardware or software failure.

3. Availability:

The software will be available only to administrator of the organization and the product as well as customer details will be recorded by him. He can add customers, update and delete them as well as add new products and manage them.

4. Security:

The security requirements deal with the primary security. The software should be handled only by the administrator and authorized users. Only the administrator has right to create new accounts and generating inventory. Only authorized users can access the system with username and password of administrator

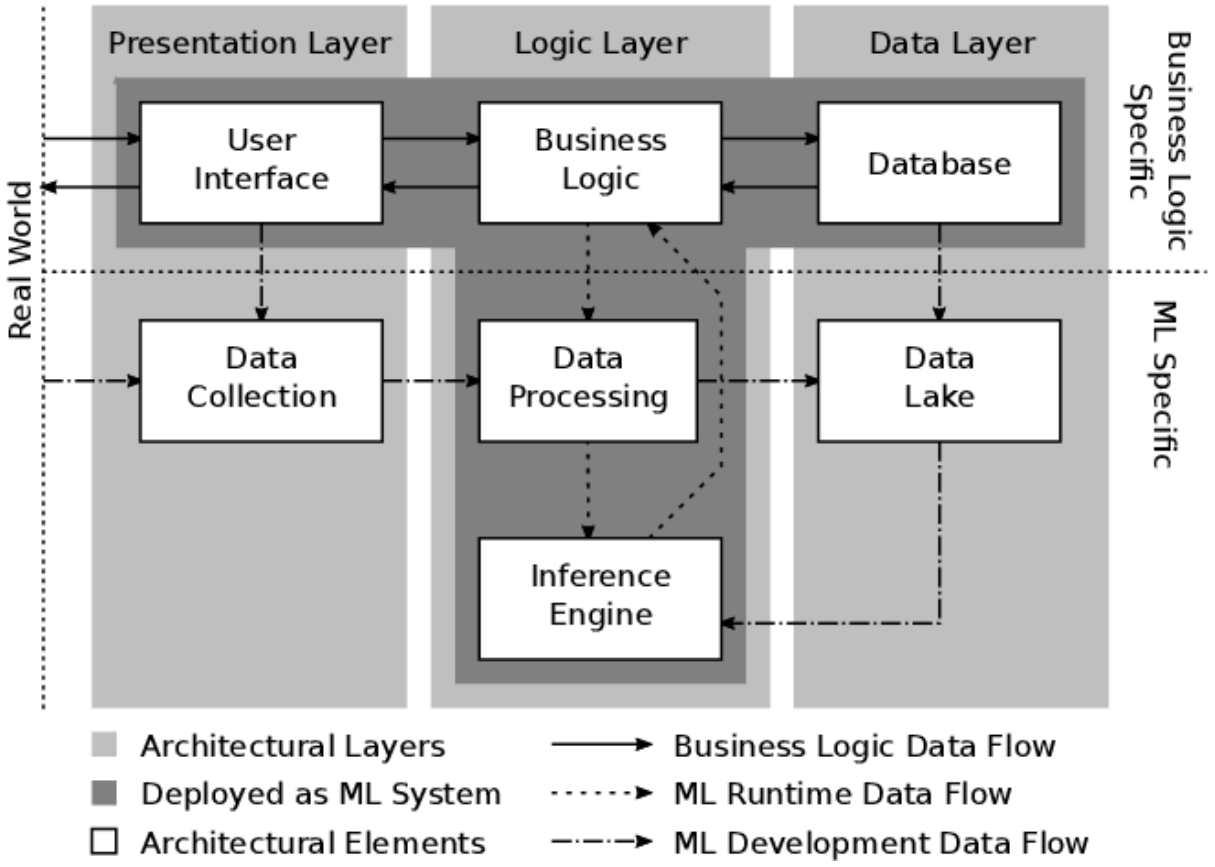
5. Maintainability:

Backups for database are available.

6. Portability:

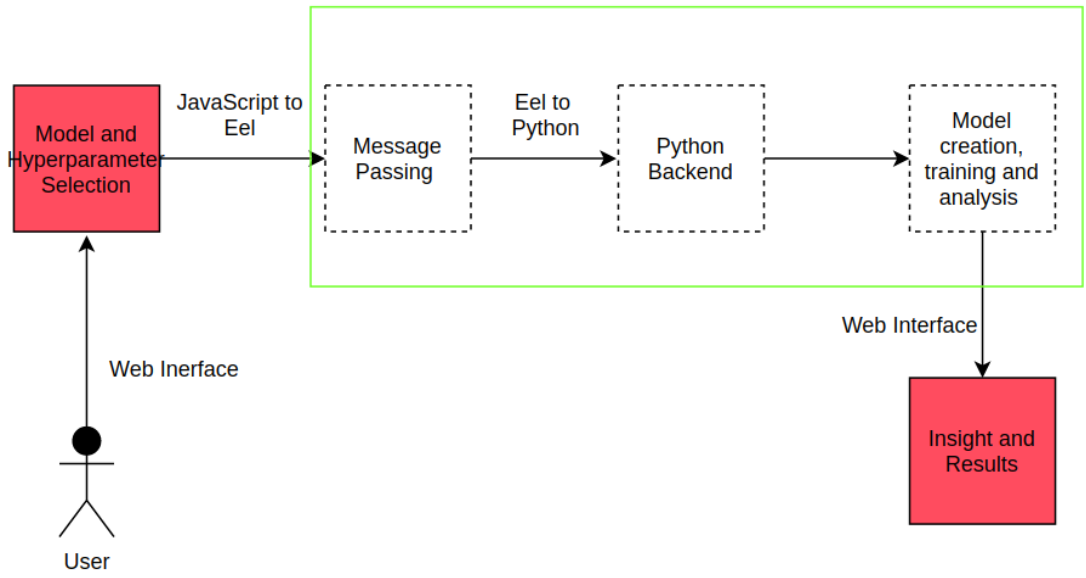
The Software is a web-based application and is built in Python and Nosql so it is platform independent and is independent of operating system.

7 LOGICAL VIEW



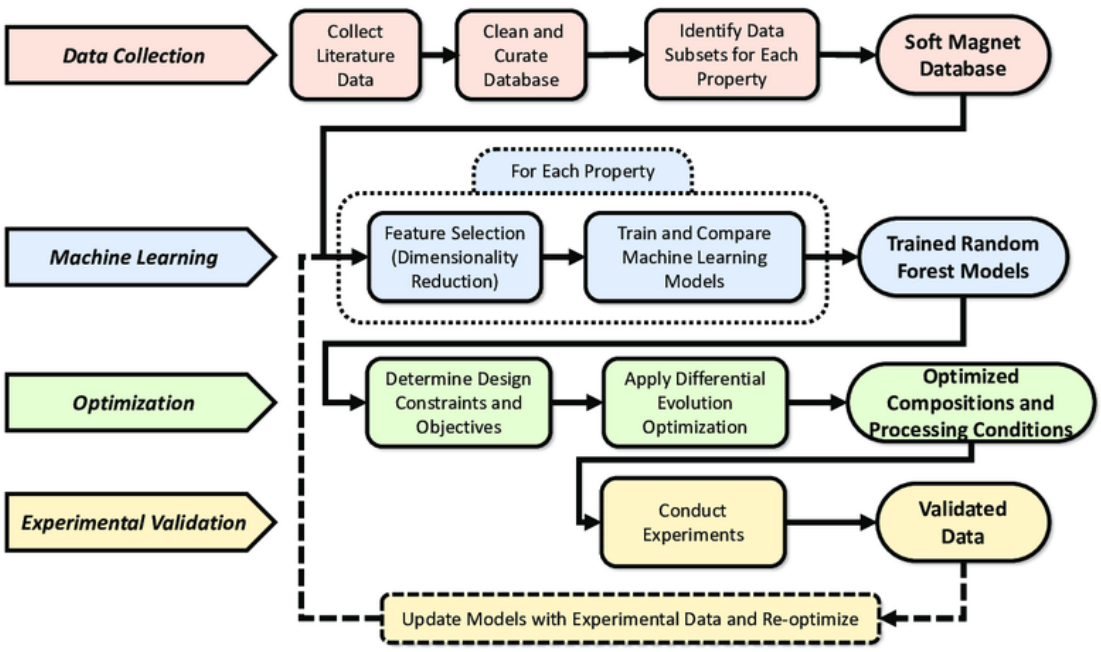
8 INTERFACE VIEW

This section describes the interfaces that will be required to the external system integration touch points

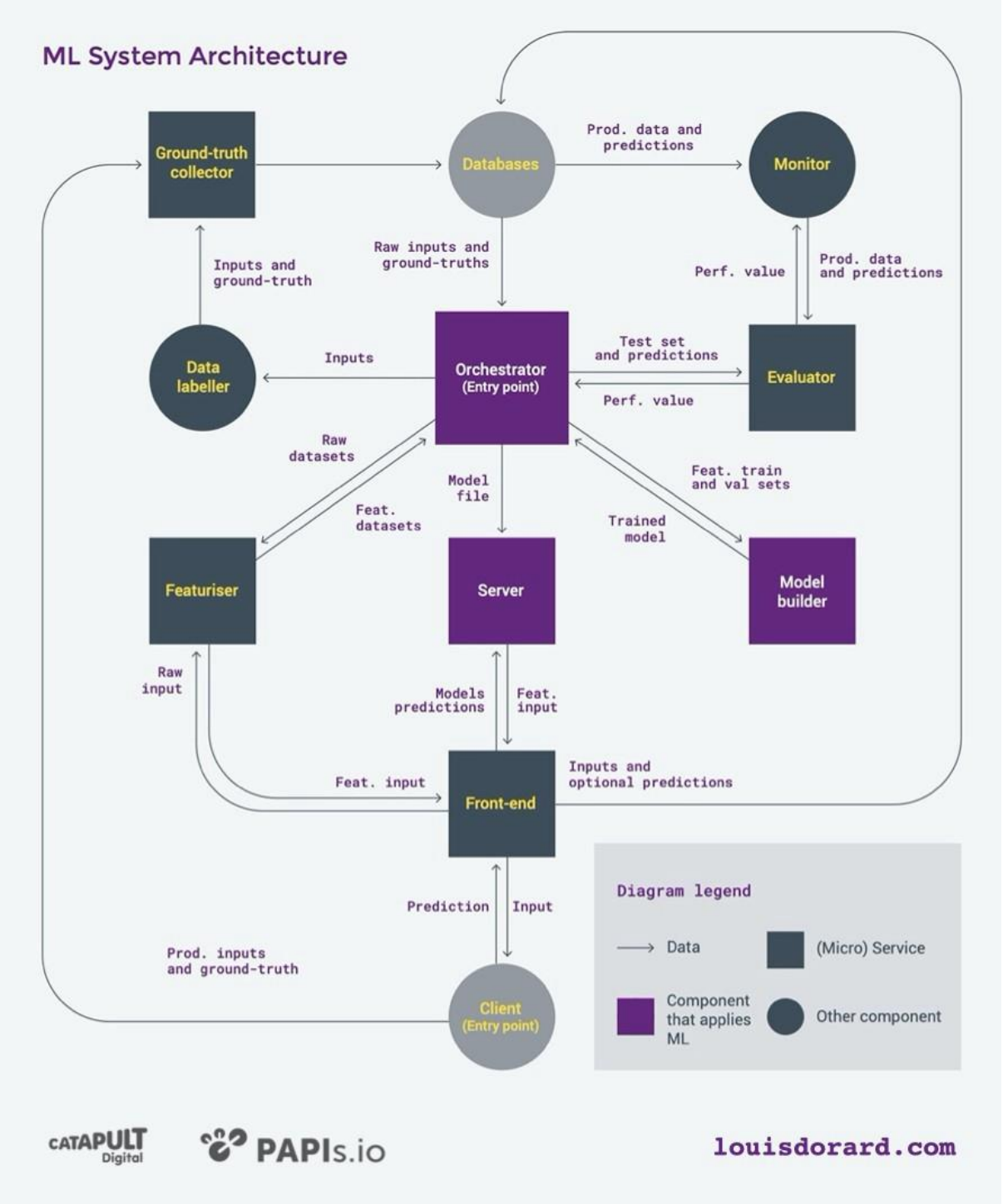


9 DESIGN VIEW

This section describes and explains any lower-level design concepts arising from the solution if required.

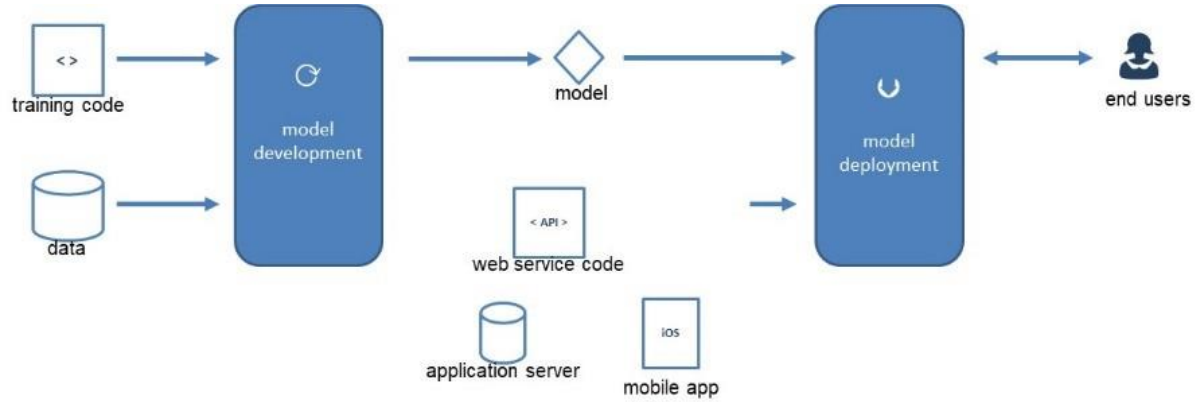


10 PHYSICAL VIEW



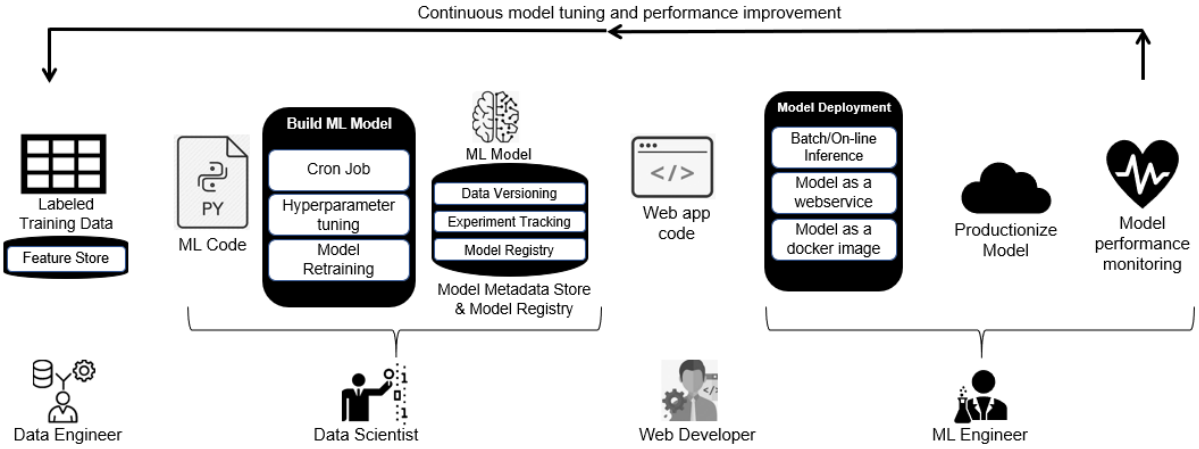
11 DEPLOYMENT VIEW

This section describes how code will be deployed in test environments and key considerations for the more complex Production go-live deployment.



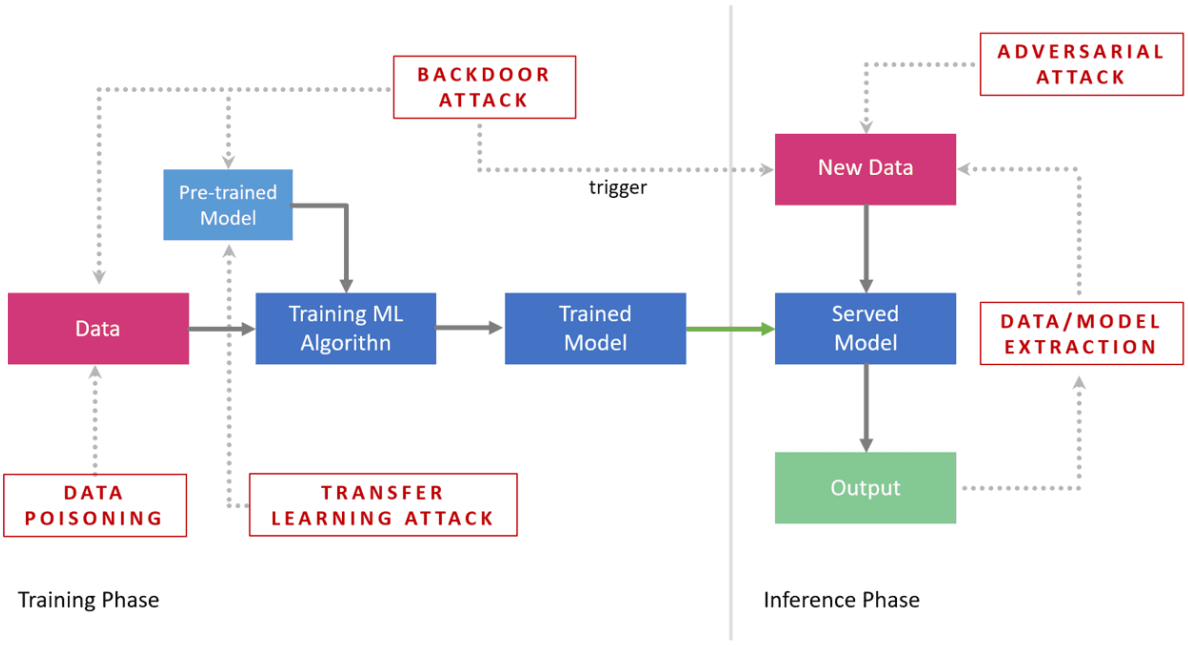
12 OPERATIONAL VIEW

This section describes how the architecture will support operational processes and activities.



13 SECURITY VIEW

This section describes how the architecture addresses the different security aspects.



14 DATA VIEW

This section describes the important data model changes required to fulfil the requirements and the associated data flows.