Requirement Document

**Group-4**

**Team 2:**

SANKALP THAKUR 2021201042

VINEET AGRAWAL 2021201049

AARADHYA GUPTA 2019114010

**INDEX**

**1. Introduction  
 1.1. Scope**

**2. Intended Use**

**2.1. Assumptions and dependencies**

**3. Subsystem Features and Requirements**

**3.1. Features**

**3.2. Requirements**

**4. Key Functions**

**4.1. Block Diagram of all components**

**4.2. Brief description of each component**

**5. Use cases**

**5.1. List what the users can do with the solution**

**6. Primary test case**

**6.1. Name of use case**

**6.2. Company executing the use case**

**6.3. Description of the use case purpose, interactions and what will the users benefit from**

**7. Subsystems**

**7.1. Interactions between modules**

**7.2. Interactions involved across these subsystems**

**7.3. External interfaces with the system**

**7.4. Registry & Repository**

## 1. Introduction

According to McKinsey Global Survey on AI,it indicates that AI adoption continues to grow and that the benefits remain significant— though in the COVID-19 pandemic’s first year, they were felt more strongly on the cost-savings front than the top line. As AI adoption grows, AI model building is also growing and therefore applications that are using AI models are also increasing.

AI model development and application development are two separate entities. So we are building a platform which connects these data scientists (who build the AI model) and the application developer.

Data scientist can upload and deploy his/her AI model to our platform and our platform makes available this model to app developers as a service, so, app developers can use these AI models by api call to his/her application.

App developer will upload/deploy his/her application to our platform and then our platform makes available this application to the end user. Then, the end user can use this application by providing details about the sensors that are going to be used by the application as an input.

### 1.1. Scope:

Our team objective was limited to deployment of models and applications as services. The models and application registrations and uploading are handled by other teams. We utilized the APIs provided by other teams to get the location of the service initiator and start the service with required details like ports and sensor details.

## **2. Intended Use:**

The services provided by the team are to be used by the request handler which takes the request by the end user for running an application. As soon as the request for application is received, the services required by the application are checked for. If they are not met, then appropriate requests are made and the application is initiated and the details of accessing the application API are returned to the user via request manager.

### 2.1. Assumptions and dependencies:

We assume that the application developer has access to the address where models and sensors are located(via the help of another platform component called DBhelper). The components we work on depend on various other components of the platform. The application deployer needs the request manager as mediator to service the request of application deployment by the user. The model deployer deploys a model when it receives a request from the application deployer. The application deployer depends on the sensor manager for binding the requested sensor to the system.

## 3. Subsystem Features and Requirements

### 3.1. Features:

#### Deployment of application:

The primary job of the subsystem is to deploy applications.The deployment request is received from the user by the request manager which forwards the request to the application deployer. After performing some secondary tasks and making sure that all the resources are available the application is deployed on one of the available ports. We assign a fixed range of ports for deployment of applications and the application deployer keeps tracks of free and occupied ports.

#### Send request for sensor binding:

When the application is deployed it needs the details of the sensors that are to be used in the application. The details of the sensor are provided by the user. These details are fetched by the application deployer and a request sent to the sensor manager containing the details of the sensor for binding the sensor. A binding id is received as a response which is made available to the application to allow the application to fetch data from the sensor.

#### Deployment of the model :

The secondary task of the application manager is to make sure that the services needed by the application are already deployed. This is done by making a request to the service lifecycle manager. Using an api call we check whether a certain model is already deployed. If not, a model deployment request is made to the model deployer for deployment of the model.

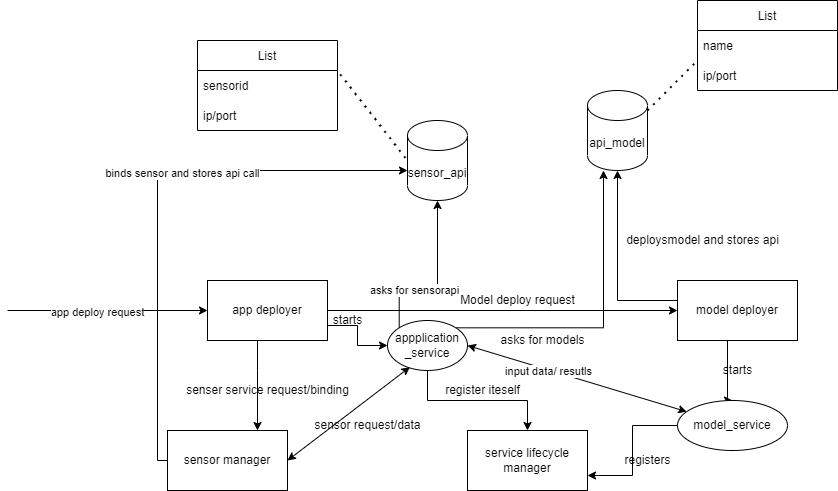
### 3.2. Requirements:

1. **Model Configuration File:** It will contain configuration and executable files related to a model following a predefined contract between Data scientist and Platform.
2. **Application Configuration File:** It will contain configuration and executable files related to a model following a predefined contract between Data scientist and Platform.
3. **UI for getting sensor details from user:** A simple UI through which the user will provide the sensor details to the application deployer.
4. **Ports:** Arange of available ports for service deployment is needed. Application services and model services will be allotted ports by their respected deployers from the range ports

## 

## 4. Key Functions

### 4.1. Block Diagram of all components



### 4.2. Brief description of each component:

1. **Application deployment** :This service is responsible for deploying the application. It is also responsible for ensuring all the resources like sensors and models are available before the app is deployed. Again, 500 predetermined ports are given to the application deployer, and the application is started on one of these empty ports. The app deployer reads the contract and finds the models that are required. Then it checks if the model is deployed or not. If the model is not deployed then it sends the request to the model deployer for deployment. It also takes the ip port of the sensor from the user and sends a binding request to the sensor manager. The request returns a binding id which is passed to the application at startup.
2. **Model deployment** : This service is responsible for deploying the model. The service is provided with 500 predetermined ports. The model deployer keeps track of these ports and deployers the model on an empty port.

## 5. Use cases

## 5.1. List what the users can do with the solution

Users should be able to use services provided by the applications (deployed on the platform) using Application ID and Application Name. Users would be provided with details of location of sensors, Number of instances of Application, etc. but implementation details would not be provided to them.

**Different user’s involved are:**

1. App Developer
2. Data Scientist
3. End User

**Data Scientist**: Their role is to create a model and deploy it on the platform which then can be used by Application Developers. They would be deploying a zip file which contains model(pickle file) and other dependencies.

**Application Developer**: Their role is to create applications using models already deployed on the platform. They would require Sensor Type and Model and would be using contract given by data scientists.

**End User**: These users would be using services provided by the application deployed(by Application Developer) on the platform using Application ID, Application Name, etc.

## **6. Primary test case**

#### 6.1. Name of use case:

Create an AI application using an ML model.

#### 6.2. Company executing the use case:

Consumer technology services company.

#### 6.3. Description of the use case purpose, interactions and what will the users benefit from:

* Machine Learning models will be generated by the data scientists, who will aggregate the data, and utilize an ML algorithm.
* Users benefit as the scientists creating the model and developers creating the applications have a common platform to speed up development.

## 7. Subsystems

#### 7.1. Interactions between modules

The following modules will be present in our Platform:

* Deployment Service
* Sensor manager
* Request Manager
* Bootstrap Mechanism
* Service Lifecycle Manager

**-Request Manager and Scheduler**

Request Manager will receive a request from the user and provide the details of the request to scheduler.This request is the end user’s request to use an application.

**-Service Lifecycle Manager and Deployment Service**

Service lifecycle manager will pass the signal(that is receiver from bootstrap mechanism) to initiate startup service to deployment service.

#### 7.2. Interactions involved across these subsystems

#### 

* Request manager can receive a request from any actor. The request is directed to its respective service or validator as needed.
* The AI runtime server fetches the model data from the Repository and returns predictions to the Request manager.
* Logging system interacts with the central server and saves logs in the log files.
* Nodes communicate with sensor through node manager

#### 7.3. External interfaces with the system

1. User interfaces

A user interface so that the user can provide details of the sensor to bind with the system. In return the user gets a binding id that is used to get sensor data.

1. Sensor interface

An interface that takes data from sensor

#### 7.4. Registry & Repository

• Registry will contain the stats of every service (what instances are running on what machines, binary files location of each service in the common file system etc.)

• Repository will contain one directory per user which will in-turn contain one directory per service and will contain all the version-revisions of that service.

.

#### 