Requirement Document

**Group-4**

**Team 2:**

ANCHAL JAKHMOLA 2021201051‌

BHUPENDRA SHARMA 2021201020

KAMAL PHOOLWANI 2021201054

**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

AI model development and application development are two separate entities. So we are building a platform that connects these data scientists (who build the AI model) and the application developer, that builds an application that will use the AI model built by the data scientists.

The data scientists will upload and deploy their AI model to our platform and our platform will make these models available to application developers as a service, so, app developers can use these AI models by API call and use them.

The application developers will upload their applications to our platform and then our platform will make available these applications to the end-user. End-user can use the applications on the network 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 to build a request manager for the platform, application upload functionality, and validator. Request Manager's objective was to route the request in the platform to proper endpoints. All the requests whether from actors or inside the component will be handled by the request manager. Application upload service provides application developers to upload their application, and the validator validates the zip uploaded by the application developer. It will validate if all the required files are present as well as contract.json inside the zip folder.

# **2. Intended Use:**

Application Upload Service will be used by the Application Developer to upload and store applications to be used by end-users. End-User will be able to see all the applications which have been uploaded and can be used. Application Developer will be providing the zip file in a specific format for which validations will be done while uploading it on the platform. While the use of Request Manager will be to take all the requests, validate and route them to respective APIs or endpoints.

## 2.1. Assumptions and dependencies:

We assume that the application developer will be providing the zip files in a specific format as mentioned in the platform.

# 3. Subsystem Features and Requirements

## 3.1. Features:

### Web User Interface:

Web User Interface provided actors(data scientists, application developers, end-user) to interact with the platform. Data Scientists and Application developers can upload their model and application using Web UI, end-users can choose the applications in the platform and use them. Web UI also provides End-Users to register the sensor details for the particular application before the use of the application.

### Request Manager:

Request Manager routes the request to proper endpoints. For example Request for uploading a model by Data Scientist is sent to the Model Upload service of the platform, similarly, an end-user request for application is sent to the deployer service which will deploy the application requested by the user and provide the instance of the application to the user.

### Application Upload:

Application upload service extracts the zip uploaded by the application developer. It will then send all the files extracted from application.zip to the validator for validating files. Once validated it will store the files and repository and update the location of the application in the application database.

### Validator:

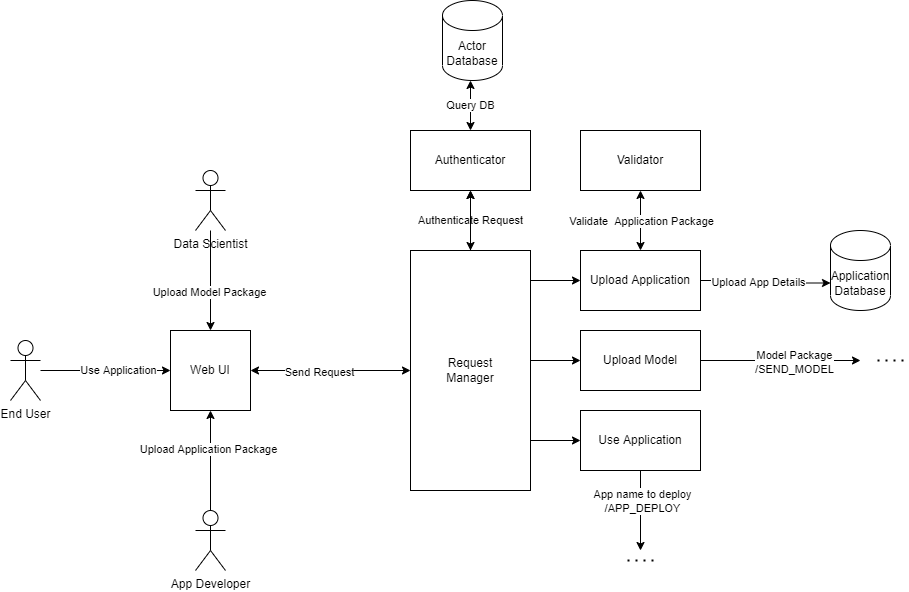
The validator will validate whether the required files are present in the application folder provided by the application developer, also validates the contract.json whether all the required keys and their values are present or not. Once validated it signals the application upload service for further processing.

### 3.2. Requirements:

1. **Model Package from Data Scientist:** For further use of Application Model package is needed in the system.
2. **Application Package from App Developer:** Application package is needed to use and deploy application
3. **UI for Interaction with Actor:** A simple UI through which the actors will provide the details to the platform.
4. **APIs:** End points are required for interaction between subsystems.

# 4. Key Functions

## 4.1. Block Diagram of all components



## 4.2. Brief description of each component:

1. **Request Manager**: This component is responsible for taking requests from 3 users that our platforms have. Users will interact with WEB UI and for each user, we have a different view present. User will end request and Request Manager will resolve request and authenticate the user and will redirect it to the relevant module. Request managers will have different endpoints responsible for different functionalities.
2. **Validator**: Validator is responsible for validating the Application package by checking its Schema. It will check the contract and validate if all possible fields are there.
3. **Authenticator**: Authenticator will get a request from the request manager for different actors and it will check if the user is valid and will respond to the request manager accordingly.
4. **Upload Application:** This component will receive an Application package, send it to the validator for validation of the package, and will upload application details in Application DB if the package is valid.
5. **Upload Model:** This component will receive a model package and will send it to the service responsible for Model validation and upload.
6. **Use Application:** This component will receive the request from the end-user for using an application. It will call Deploy app API by providing the app name.

# 5. Use cases

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

Users can directly interact with WEB UI and can perform relevant tasks they are intended to.

**Different user’s involved are:**

1. **App Developer** : App Developers will have to authenticate themselves and will get a view which allows them to upload the zip file for their application.They will receive a response(error or success) based on validator processing.
2. **Data Scientist** : App Developers will also have to authenticate themselves and will get a view which allows them to upload the model which will be further unzipped, validated and uploaded.They will receive a response(error or success) based on validator processing.
3. **End User** : End user will have to authenticate and then they will be able to use any application which is uploaded in Application DB.

# **6. Primary test case**

### 6.1. Name of use case:

Upload an AI application on our platform.

### 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**

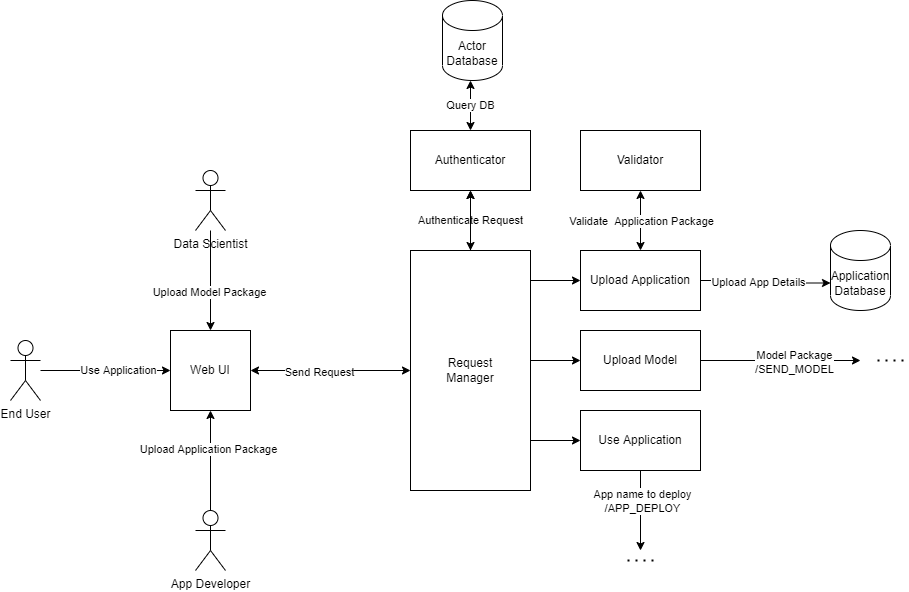
Request Manager will receive a request from the user and provide the details of the request to the scheduler. This request is the end user’s request to use an application.

**-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

* **WEB UI** will interact with **Request Manager**,all the requests from actors will be passed from WEB UI and will enter into the system from Request Manager.
* The **Request Manager** will interact with the Authenticatorto authenticate the actor.
* The **Request Manager** will also authenticate with 3 components **App Upload,Model Upload** and **Use App** by passing relevant data.
* **App Upload** will interact with **Validator** to validate the application package.



#### 7.3. External interfaces with the system

1. User interfaces:

WEB UI will interact with Request Manager,all the requests from actors will be passed from WEB UI and will enter into the system from Request Manager.

1. Request Manager:

Request Manager will receive a request from the user, validate and route them to respective APIs or endpoints.

#### 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 to store extracted folders corresponding to each application from zip files during upload.

#### 