1. Overview

a. Introduction

i. Definition what constitutes in this project 1-3 para

The goal of this project is to create a unified platform consisting of microservices that can be used to package, deploy, run, monitor and manage AI based models and applications that utilize these AI models. It should also support the ability to configure and manage distributed sensors that provide data and controllers which can be used to control other physical devices.

In other words, the platform will provide an environment for the users to connect to / use location sensitive sensors, controllers and also AI models from their own applications. It will provide an interface in the form of a dashboard which will expose the utilities in a user-friendly way. Data scientists can deploy their AI models in this platform in a secure manner. They can also configure the deployment (for eg. they can choose the number of instances of the app to be deployed, the environment of the server where it would be deployed etc.). After the models are deployed, the end app developers can use these hosted models in their applications and after the application is created and built, they can use the platform to deploy it in a secured and configured manner as well. The end app developer can also use the registered sensors and controllers on the platform in their application to fulfill their application's purpose.

The deployed instances of the models and applications on the platform would also be constantly monitored and tracked using monitoring, health check and logging services. The usage of the server machines containing the deployed instances would be made secure using an authentication and authorization service.

b. Scope

The scope of this project can be summarized in the following points:

1) The platform implemented in the project should be able to support the deployment and monitoring of the following types of applications:

    a) AI based applications that use a trained model.

        - A single application can use two or more different models available on the platform.
        - It is possible that the output of the first model can be the input of the second model.

    b) Application endpoints hosting input data based on sensor readings.
        - Sensor data can be of the following types:
            - Image
            - Video
            - Numbers
            - Strings

    c) Application endpoints hosting trained models.

2) The platform should provide some interface exposing the APIs that the user can use to access its functionalities.

3) The platform should provide contracts containing information on how to deploy the models/applications. It should be able to handle the deployment based on this provided contract.

4) The data scientist will also be uploading a contract onto the platform that contains information about how to use the model along with their model instance and the platform should be able to expose this contract with the served model.

5) The platform should provide functionalities that can be used to add configuration about the sensors and controllers.

6) There should be resource management capabilities in the platform to manage the limited resources.

7) The platform should be:
    a) Distributed (using microservices)
    b) Fault Tolerant
    c) Secure
    d) Scalable

8) Load balancing and resource balancing should be supported.

9) Both the AI model developer and the end application developer should be able to configure the deployment of their models or applications using a configuration file that they can pass to the platform during deployment. The types of deployment configuration supported by the platform could be the number of instances of the application that would run on the nodes, the base environment on which the application would run etc.

10) Several microservices like the authorization service and the health check service can be created and used to secure and monitor the deployed applications and models.

11) The end user should be able to use any application or sensor data by providing the application id or name, or location of the sensor but he should not be aware of the implementation.

2. Intended Use

a. Intended Use

The platform would provide faster, more efficient, more effective collaboration with Data Scientists, application developers and the end users. It would act as a one-stop solution.It can help minimize costs in a number of ways – preventing duplication of effort, automating simple tasks.

b. Assumptions and dependencies

- The end app developer and the data scientist would provide appropriate contracts that can be used for proper functioning of the models and applications after deployment.
- The data scientist would provide a sufficiently trained model for deployment in the platform as the platform does not provide training and testing facilities for AI Models.
- The ids of the sensors and controllers used in the applications would be provided in the contract by the end application developer.

3. System Features and Requirements

a. Platform Requirements

i. Deployment of the platform

The platform should provide capability of deploying the supported types of applications onto servers that are catered to a specific configuration.

ii. Different actors on the platform

1. **Data Scientist** - Makes the AI model, should be able to package and upload the model based on contract provided by platform. Platform will avail it as service for applications on the platform.
2. **Application developer** - Develops the application based on the contract provided by platform on how to package application and configuration files. Application will use models available based on contracts provided by data scientists. App will be based on models, sensor type, controller type available on platform.

3. **Platform Configurer** - Configure the sensors and controller on the platform.
4. **End Users -** Tells type, number and location for sensors

iii. Applications overview of the platform

b. Functional Requirements

i. AI Model

1. Development of AI Model

The AI model which will be developed by the application developer outside the platform.

2. Packaging of Model

Data Scientist needs to package preprocessing and postprocessing functions in the configuration file according to the contract provided by the platform.

3. Upload and deployment of model

The Data Scientist should be able to upload/deploy the packaged AI Model onto the platform. The platform should allow configuring the deployment using a config file.

ii. Application

1. Registering sensors

Sensor registration is responsible for initial setup of sensors and binds the sensors to a gateway for dumping data.

2. Interaction with sensors

The sensors that are registered in the platform can be accessed using APIs. When we pass the correct sensor id of the sensor to the API it will return the data corresponding to that sensor.

An application developer who wants to use a sensor can include the sensor type and the sensor location in his application's configuration and the platform should handle the interaction between the application and the sensor. This can be managed by a sensor manager module.

3. Development of application on the platform

The development of the application would be handled outside the platform. The platform is only responsible for deploying and maintaining the deployed applications.

4. Identification of sensors for data binding

The sensors will be identified by the type, location and other metadata provided at the time of registration. Sensor manager will get a request of sensor data from init.py file running at some node. For each such request, the sensor manager will detach a thread to serve the request.

5. Data Binding to the application

The thread serving the request will be responsible for reading sensor data from sensor topic, preprocessing the data as per config file and write the data to temporary topics at a given rate provided by the sensor manager.

iii. Scheduling on the platform

The Scheduler is mainly responsible for sending the jobs to the deployer queue. We can have a user interface to accept the application to be deployed in a specified format which is then stored into the queue. Each job in the queue is assigned a priority depending on various factors such as the resources it requires, or the computation. A job is then picked from the queue and is scheduled for a specific time. This job is then sent to the deployer for execution. A scheduler can also specify the end time, that represents the time when the scheduler wants to kill its execution

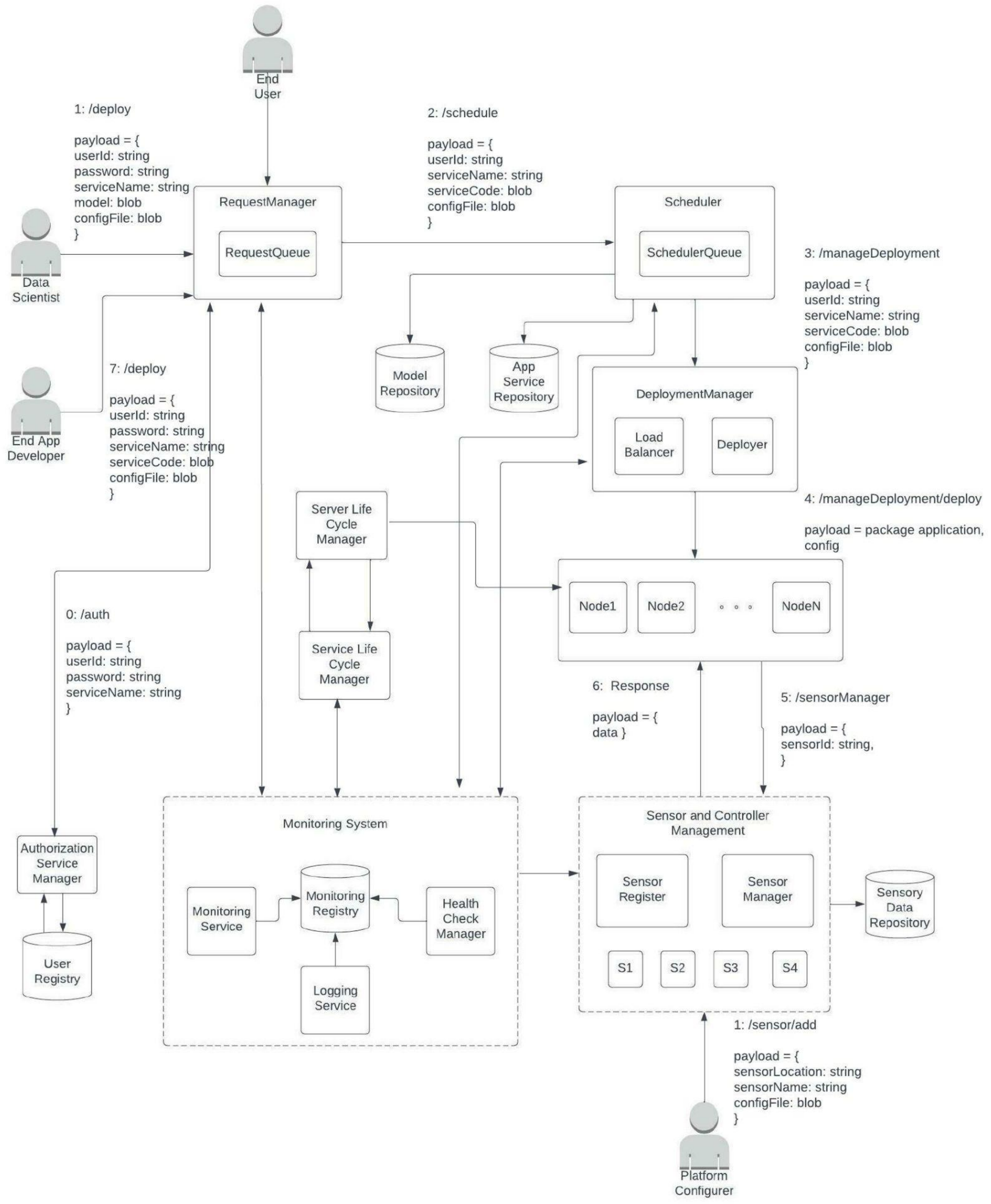
iv. Acceptance of scheduling configuration

A configuration file can be attached with the application that needs to be deployed. It will contain the metadata of the application and the application developer. We also need to specify the scheduling dependencies that include the services that are required to be run before deploying the application. Also, the order in which these dependencies are expected to run.

v. Starting and Stopping services

The config file shared by init file will specify at what time to start serving data and when to stop.

vi. Communication model:at



vii. Server and service life cycle

The service lifecycle service requests the server life cycle service to start a new server node whenever a new service is deployed. If a new server is started successfully then the server lifecycle module sends the ip and port information of the new server back to the service lifecycle service.

viii. Deployment of application on the platform

The deployer sends the request for deploying the application to the platform which is saved in the queue. This job is picked by the scheduler. The Scheduler analyses the configuration information and the algorithms to get a better understanding of the application. It then specifies the start time and duration for which it should run.

The deployer demands for a single or more modes to deploy the request. The application is then sent for deployment on the platform and a load balancing algorithm is applied to distribute the workload. The deployer will then have to monitor the health of each node.

ix. Registry & repository

A Registry will be used to store the run time information of the application modules. Such as storing the login details in the database when a new user login to a module. And storing the IP addresses and the ports of each node on which the application instance is running.

A Repository will store the static files of all the applications such as the configuration file containing the metadata of the application and its modules.

x. Load Balancing

As the application request is received by the deployer. It gets the free nodes from the platform for deployment. A load balancing algorithm is then applied, and the active workload is calculated for each of the available nodes by noting the CPU usage and the RAM usage of each. In case of high workload, the load balancer requests for more nodes from the manager.

xi. Interactions between modules

The deployer sends a request for deployment of an application on the platform which is put into the waiting queue. The Scheduler picks up the job from the queue at the execution start time. It analyses the dependencies and forwards the request to the Deployment manager. The deployment manager receives the execution request and applies a load balancing algorithm. The node manager is responsible for initiating nodes and for communication between the load balancer and sensor management. The Sensor Manager communicates with the Deployment Manager using init file and is responsible for sensor registration, processing sensor data, controlling sensor data rate and sending sensor data. The monitoring system is then responsible for monitoring the modules and services deployed. In case of abnormality in behavior of any module, it requests the Fault Tolerant Manager to investigate the issue and take necessary action.

xii. Packaging details

xiii. Configuration files details

The platform will receive a configuration file with the application that contains the application name, Id and the dependencies associated with it. It must also contain the details of the sensors that it requires such as the type and number of instances of the sensor required and the action to be performed by them. It contains all the data to make the developed application compatible with our platform.

xiv. Interaction of different actors with the platform

1. **Data Scientist -** Packages model according to contract provided by platform and deploy it on the platform.
2. **Application Developer -** Develops apps according to contract provided by platform and deploy it on the platform, and based on model, sensor type, controller type available on platform. Application will use API provided by platform to get data from sensors and send signal/commands to controller.
3. **Platform Configurer -** Configure and register the sensors and controller on the platform.
4. **End Users -** Tells type, number, scheduling details and location of required sensors.

c. Non-Functional Requirements

i. Fault tolerance

1. **Platform**: The Monitoring system will monitor whether all modules are running correctly. If there is any issue in any module behavior, it will request the fault tolerant manager to look into the issue. The Fault tolerant manager will

see if the issue is with the node or with the module itself. If the problem was generated in the node, it will run the module on a different node.

2. **Application**: The deployed application can be modified by the entitle developer and hence in case of any failure it could be fixed and redeployed.

ii. Scalability

1. **Platform**: Scalability will be provided on the platform by allowing more than one instance of an application to run concurrently.

2. **Application**: The deployed applications could be re-deployed after scaling or extending it by the entitled developer.

iii. Accessibility of data

1. **Application**: The applications can directly use the data provided by the sensors and control the controllers using signals by adding the details of the sensors and controllers in the configuration file. The platform will handle the interaction between the applications and the sensors using the information provided in the configuration file.

2. **Sensors**: The data from the sensors could also be directly retrieve using API endpoints.

iv. Specification about application

The platform will support all applications that contain models trained using any machine learning algorithm.

v. UI and CLI for interaction

We will create a user interface and provide command line interaction for the user to deploy the application, set priority of the service, provide credentials for logging in, etc. Also, to monitor the applications running.

vi. Security - Authentication and Authorization

Authentication is to allow the user to enter its username and password and our platform needs to verify it from the database.

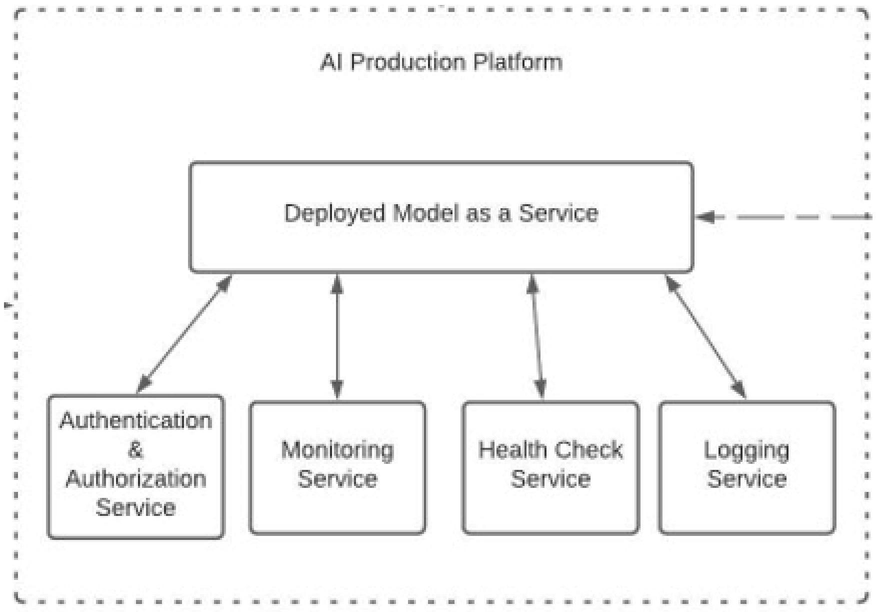
Authorization is to check if the particular user has the permissions and the access rights for using the service.

vii. Persistence

The state of the platform that includes the state of applications, schedules and the sensors is saved when the platform shuts-down and is restored when the platform starts up again.

4. List the key functions

a. A block diagram listing all major components ~components



b. Brief description of each component

1. **Authentication and Authorization:** Takes the username and password as input and verifies it from the database and returns the result. Also, it will check for the list of services that are accessible to the user.
2. **Monitoring Service:** The monitoring system is responsible for continuously monitoring the status of all the other modules. It is also responsible for verifying if an instance of the module itself is working or not. It detects and reports any abnormal behavior of any module and the node on which it is running.

3. **Health Check service:** The Health Check Service is the load balancer. It checks the health of each module by periodically sending in the requests, attempting connection, or sending a ping to them and getting back the status. It distributes the workload among all the running instances of the service.
4. **Logging Service:** It is a log monitoring service that checks if the log file content has been updated or not. It can take the log file path and the log message as the input and check whether the service has written the specified log message onto the log file or not.

c. List the 4 major parts each team will take one part

- Scheduler
- Deployment Manager
- Monitoring System
- Sensor Management

5. Use cases

a. List what the users can do with the solution

1. Data Scientists can deploy their model, and can reach a wider set of app developers, making them use their model.
2. App developers can connect multiple machine learning models to their application.
3. App developers can schedule and deploy their application on the platform. There can be multiple applications running at the same time, but they will be isolated.
4. Can enable app developers to make better decisions based on the predictions made by our models.
5. Naive app developers can have their applications have AI enabled features.
6. Platform configurers can configure different types of sensors and controllers on the platform.

b. Who are the types of user's name, domain/vertical, role, what they are trying to do when they need the system?

User Type	Vertical	Use Case
Data scientists	Researchers	Data scientists can upload any model on the platform and enable it to be accessed by a wider set of developers.
App Developers	Student/ Engineers	App developers can enable their applications to have access to state of the art machine learning algorithms.
End User	Novice/ Professionals	End user gets to access our services through the application that has been deployed on our platform

c. At least 5 usage scenarios. Give name, and a brief 3-5 lines description.

- **Smart-Home Setup**
  App developers can use our platform to make their smart homes related applications make smart decisions based on data collected by various sensors and our AI models.

- **Traffic Violation Detection:**
  Any traffic violation system might require to detect the license plate of any speeding vehicle or any other rule breaker. He can configure a camera sensor to send a continuous stream of video from which our platform will detect the violation and report back to the authorities the license plate of the suspect.

- **Face Mask Detection:**
  Any institute or organization can use our platform to configure an application to detect persons not wearing facemasks within the premises.

- **Face Recognition:**
  Any application requiring biometrics related authorization, can use our platform to automate the process of authentication.

- **Character Recognition:**

Any application developer might require his app to have the ability to recognize handwritten text in PDFs/documents can use our platform to automate the process of extraction.

6. Primary test case for the project that you will use to test

a. Name of use case

Face Recognition Based Security System

b. Domain/company/environment where this use case occurs

Offices, Banks or any place where we need to authenticate a person.

c. Description of the use case purpose, interactions and what will the users benefit

An organization might require to restrict access to certain protected resources like computer systems, networks, databases, websites and other network-based applications or services to authenticated users(or processes) only.

In such cases our platform can help them automate the process authentication onsite/remotely.

d. What is the “internet: angle around these things”:

Users can be authenticated over the internet, this can enable user to access shared resources over the network.

Sensor will send a set of pictures of the user to the application hosted on our platform. Application will verify user using models available on our platform and signal the respective controller to take appropriate action.

Communication between sensors/controller and application/platform will be done using REST API using HTTP requests.

e. Information model

- End-user will initiate the request.
- Application will call APIs to make senor collect required data.
- Sensor will collect the data and send it to the application hosted on our platform
- Application will then access our AI model to make decisions.
- Based on that decision, application will send signal to the controller to perform the required action.

f. How is location and sensory information used

Authenticate/verify users to decide whether to give access or not.

g. Processing logic on the backend

Application will process user requests, call appropriate APIs to get data from the sensors. After receiving the data our application will extract the face from the photograph and preprocess it to send to the AI Model according to the contract provided by the data scientist.
Based on decisions made by the AI model, our application will make API calls to the controller to take the required action.

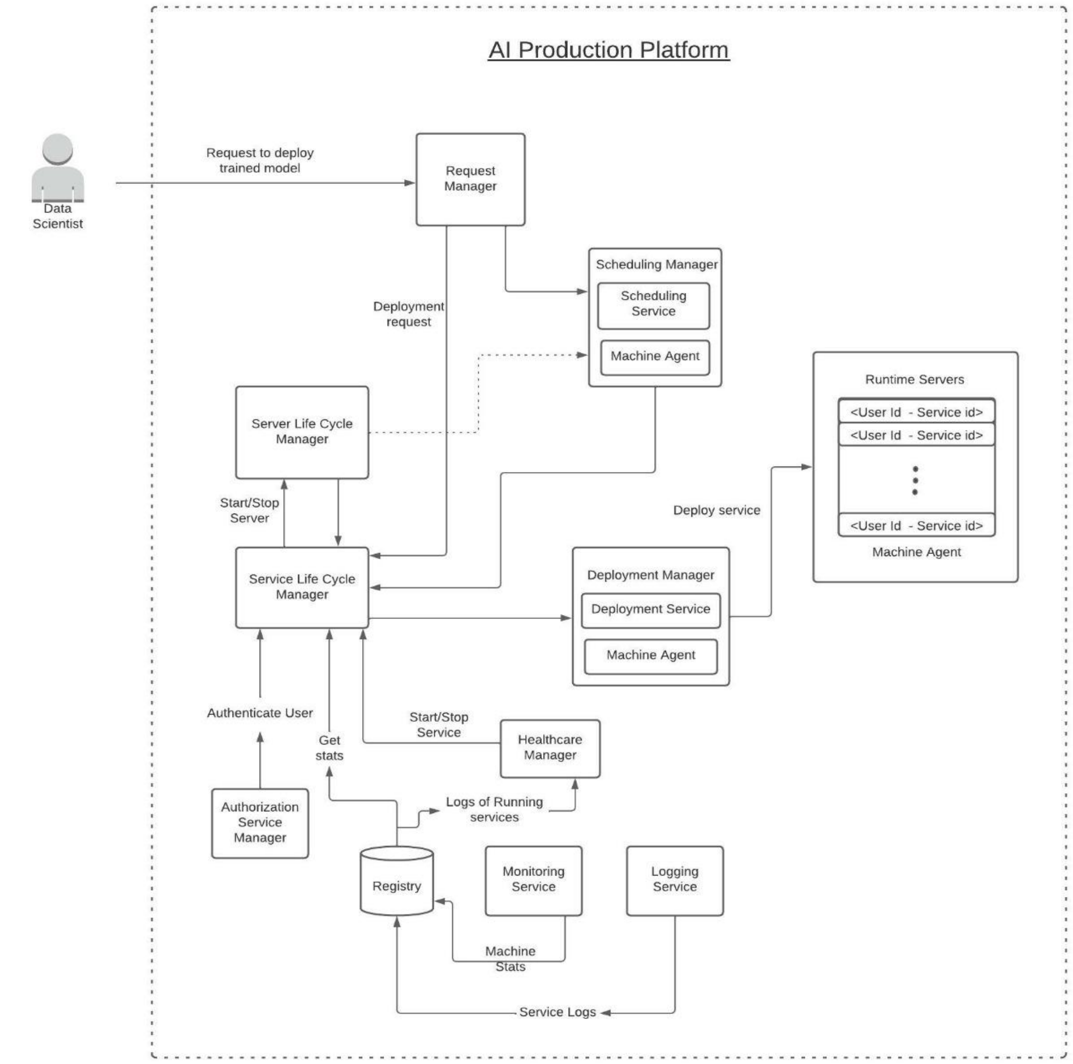
h. **User's UI view**- what is their UI, where and all will they do with these systems
The application developer and data scientist will be having a Graphical User Interface for deploying applications and models respectively.

7. Subsystems

a. Key subsystems in the project

01. Platform Manager
02. Sensor manager
03. Communication module
04. Monitoring and Fault tolerance Manager
05. Deployment manager
06. Scheduler
07. Service Lifecycle Manager
08. Request Manager
09. Authorization Service Manager
10. Runtime Servers

b. A block diagram of all subsystems



c. Interactions involved across these subsystems

- **Platform Manager** -Platform manager manages the configuration requirements for all the different platform services and deploys them on servers . It is responsible for providing the contracts for the deployment of various models along with the management of the instances of the sensors, AI models and application models.
- **Monitoring and Fault Tolerance Manager -** Monitoring module receives status updates, heartbeat pings and logs from all the modules and fault manager uses this information to reschedule failed services, applications and services.

- **Scheduler -** The scheduler is responsible for managing and passing the details of the scheduling services , which are further passes to the Service lifecycle manager and subsequently to the deployment manager.
- **Sensor and Manager** - Sensor Manager accesses the device's sensors. It interacts with the various sensors and controllers and passes it to the sensory data repository.
- **Service Lifecycle Manager** - The service lifecycle manager acts as the authority for communicating with the monitoring system. It also passes the information to various nodes which are managed by the deployment manager.
- **Authorization service manager** - It would authenticate the users by interacting with the user registry.
- **Runtime Servers** - The runtime servers would get the deployment services related information from the deployment manager and they would keep track of the users and their corresponding service id's.
- **Deployment Manager** - deployment manager uses the scheduling information to deploy applications on servers.
- **Communication module** - The communication manager manages the different components of the system . It communicates with each other and transfers the requests and responses among them based on some communication standard.

d. Protocols. Mechanisms.

e. External interfaces with the system

f. Registry & Repository

A Registry will be used to store the run time information of the application modules. Such as storing the login details in the database when a new user login to a module. And storing the IP addresses and the ports of each node on which the application instance is running.

A Repository stores the static files. We can have two repositories for storing the static files related to each application separately in an Application Repository that will contain the configuration files, etc. And a Platform Repository that will store the network files and the files required for setting up the platform initially.

g. The four parts of the project each will have its own team req. doc to be submitted.
- Scheduler
- Deployment Manager
- Monitoring System
- Sensor Management

8. Brief overview of each of the four parts

a. What are the four parts each team will work on one part

- Scheduler
- Deployment Manager
- Monitoring System
- Sensor Management

b. For each part

c. Functional Overview of each part

- **Scheduler:** The scheduler is mainly responsible for sending the jobs to the deployment manager for execution. It will receive the metadata about the application from the developer and store it in the scheduling queue. Each job in the queue is then assigned a start time for execution and the end time.

- **Deployment Manager:** The deployment manager is responsible for deploying the application or the service on the machine. It will receive the configuration file of the application to locate and deploy all the dependencies. It is also responsible for applying the load balancing algorithm and initiating the nodes on the platform to run the service.

- **Monitoring System:** The monitoring system and fault tolerance module is responsible for continuously monitoring status of all other modules like the Scheduler, Deployment Manager etc and their corresponding nodes on which they are running. If the system detects any abnormal behavior or error in the platform, it checks whether the module showing abnormal behavior is not working or the nodes on which the module is running are not working, and takes necessary steps as required.

- **Sensor Management:** The deployed application will use sensor data to produce output. The sensor manager will be responsible for preprocessing sensor data, binding sensor data with deployed algorithms, controlling sensor data rate and sending data as per the configuration provided by the application developer.

d. One block diagram giving more details on this part

- Scheduler

- Deployment Manager

- Monitoring System

- Sensor Management

e. List of sub-systems in this part

- **Scheduler**
    - Job Execution
    - Job Termination

- **Deployment Manager**
    - Deployer
    - Node Manager

- **Monitoring System**
    - Monitoring Module
    - Fault Tolerant Manager

- **Sensor Management**
    - Sensor Registration
    - Sensor Manager

f. List of services/capabilities in the part

- **Scheduler:**
    - Store incoming requests in the queue
    - Assign execution Start time
    - Assign execution End time and perform the schedule kill operation

- **Deployment Manager**
    - Deploy the algorithm and its dependencies on the platform
    - Initiating node instances
    - Load Balancing

- **Monitoring System:**
    - Monitor status of other modules
    - In case of abnormality in behavior of a module, request Fault Tolerant Manager to handle the issue

- **Sensor Management:**
    - Sensor Registration
    - Processing sensor data
    - Control sensor data rate
    - Sending sensor data

g. Interactions between this and other parts. Nature/purpose of interactions, likely interchange info/services

- **Scheduler:** The Scheduler picks up the job from the queue at the execution start time. It analyses the dependencies and forwards the request to the Deployment manager.

- **Deployment Manager:** The deployment manager receives the execution request. The node manager is responsible for initiating nodes and for communication between the load balancer and sensor management.

- **Monitoring System:** It monitors the modules and services deployed by the Deployment Manager. In case of abnormality in behavior of any module, it requests the Fault Tolerant Manager to look into the issue and take necessary action.

- **Sensor Management:** It communicates with the Deployment Manager using init file and is responsible for sensor registration, processing sensor data, controlling sensor data rate and sending sensor data.