INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY

HYDERABAD

IOT Based Application Platform

Project Requirements

IAS - Spring 2023

Group - 05

1. Overview :

a. Introduction :

With growing IoT use, respective device data is also growing at a similar rate. Each sensor sends potential data which can be used to make an important decision. The difficulty arises for a developer to keep track of this sensor data and use this data as input in its application. Managing algorithms, binding data, and analysing the output are some big tasks a developer has to take care of. The aim of this platform is to provide the user with an environment that allows them to create applications which can connect to / use location-sensitive IoT devices with ease. In addition to this, it also provides the capability to dynamically run user-defined algorithms which make use of such devices. This IoT Application platform directly integrates with the IoT devices and is capable of performing remote actions on these devices.

b. Scope :

Integration: IoT platforms often provide APIs and other integration tools to enable the platform to be connected to other systems and applications.

Security: An IoT platform ensures the security and privacy of data generated by IoT devices. It includes features like authentication, encryption, and access control.

Application Development: An IoT platform provides tools and frameworks for building custom IoT applications, enabling faster development and deployment.

Connectivity Management: An IoT platform facilitates communication between devices and enables seamless integration with third-party systems.

Data Collection: IoT platforms collect data from devices and sensors, and store it for further analysis.

2. Intended Use :

a. Intended Use :

The intended use of an IoT platform is to facilitate the deployment, management, and scaling of IoT solutions.

b. Assumptions and dependencies :

No Human Intervention: It means that the system is fully automated and does not expect any human interaction to function properly.

Third Party Services: It is expected that the third-party services employed by the platform are acting in accordance with expectations.

Device Compatibility: It is essential to assume that the platform will be able to communicate and interact with any device that complies with standard communication protocols such as Bluetooth, Wi-Fi, Zigbee, or MQTT.

Connection: Reliable internet connectivity is required for Sensors Nodes/ IOT devices to communicate with the platform.

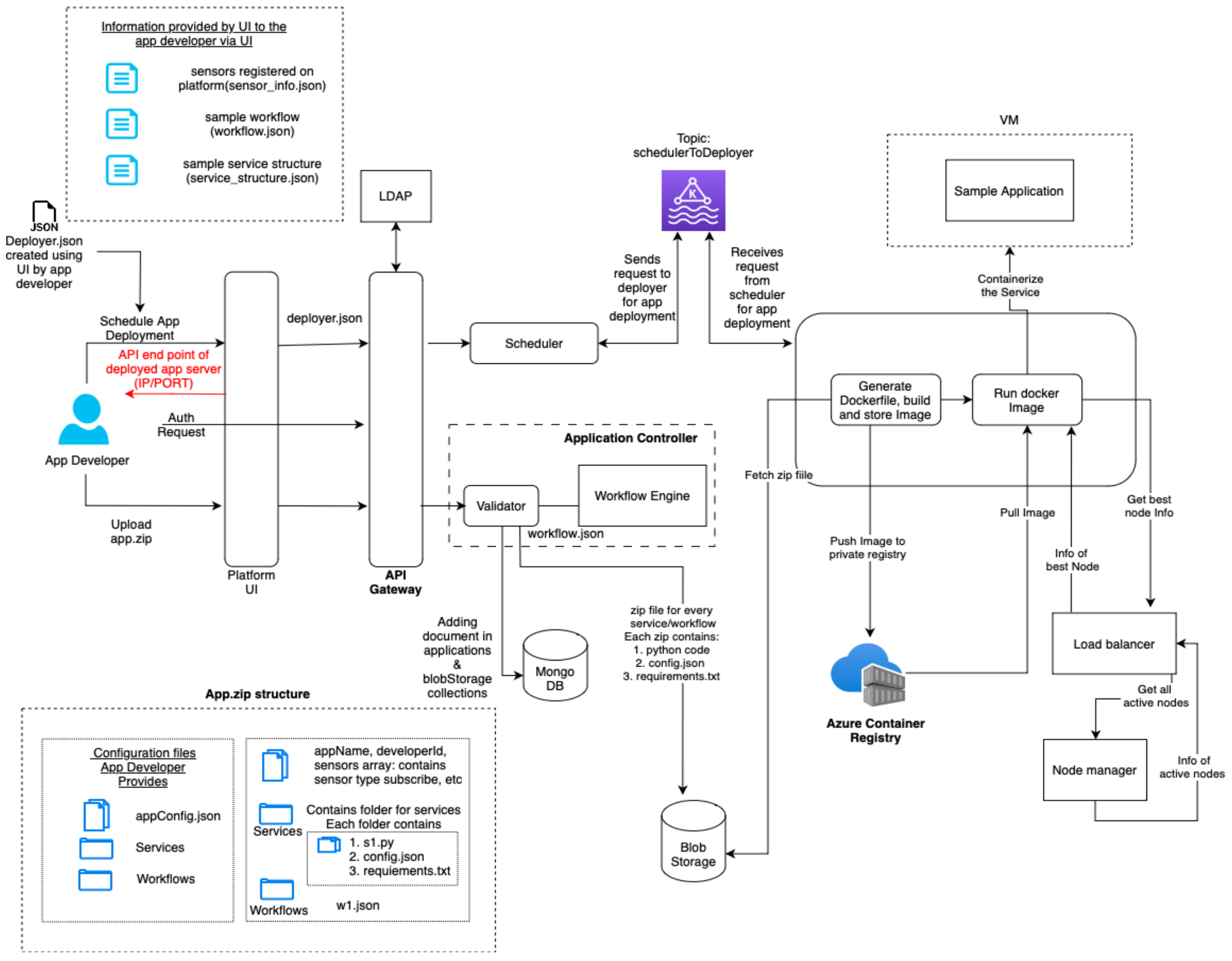
3. System Features and Requirements :

a. Platform Requirements :

i. Different actors on the platform :

End User
Application developers
Platform Developers
Platform Admin

ii. Applications Model: Uploading & Deployment


Information provided by UI to the app developer via UI
sensors registered on platform(sensor_info.json)
sample workflow (workflow.json)
sample service structure (service_structure.json)
JSON
Deployer.json created using UI by app developer
LDAP
Topic: schedulerToDeployer
VM
Sample Application
Schedule App Deployment
API end point of deployed app server (IP/PORT)
deployer.json
Auth Request
App Developer
Upload app.zip
Platform UI
API Gateway
Scheduler
Sends request to deployer for app deployment
Receives request from scheduler for app deployment
Containerize the Service
Generate Dockerfile, build and store Image
Run docker Image
Application Controller
Validator
Workflow Engine
workflow.json
Fetch zip fiile
Pull Image
Get best node Info
Push Image to private registry
Info of best Node
Adding document in applications & blobStorage collections
Mongo DB
zip file for every service/workflow Each zip contains: 1. python code 2. config.json 3. requirements.txt
Azure Container Registry
Load balancer
Get all active nodes
Node manager
Info of active nodes
Blob Storage
App.zip structure
Configuration files App Developer Provides
appConfig.json
Services
Workflows
appName, developerId, sensors array: contains sensor type subscribe, etc
Contains folder for services Each folder contains
Services
1. s1.py
2. config.json
3. requiements.txt
Workflows
w1.json

Application Model

b. Functional Requirements :

i. Application :

1. <u>Sensor Registration :</u>

- Sensors will have a unique ID through which each sensor can be uniquely identified.
- Sensor manager accepts sensor config file from Application Controller.
- For each service of the application, the required number of sensors of the specific type is mapped with the service and a Kafka channel is created.
- The list of created Kafka topics is sent to the Application Controller.

2. <u>Development of application on the platform :</u>

Development Model Steps:

- **Step 1:** Platform UI(Platform Manager) will provide the information to the Application Developer.
  Eg: File Format, Sample service, Sample Workflow, Sensors Registered etc
- **Step 2:** Application Developer creates a zip file which contains script and config files from the above provided information.
- **Step 3:** Application Developer then uploads a zip file via Platform UI.
- **Step 4:** The zip file provided by the application developer is then validator validates the zip file (i.e whether the zip files contains all the files necessary and the structure of the config files etc) and generates the workflow if any and stores the services and workflow zip on the azure blob storage and its corresponding details in mongodb.
- **Step 5:** Validator Workflow subsystem passes the sensor config data to the Sensor manager
- **Step 6:** Sensor Manager validates the sensor configs data,and after successful validation allocates sensors filtered by sensor type.

3. <u>Providing Live Sensor Data:</u>

- Live sensor data fetched by OneM2M API is produced in the allocated Kafka topic for each registered service.
- Sensor Data is stored for each sensor instance in the Sensor Repository for data analysis.

4. <u>Data Binding to the application :</u>

- An Application instance running on a node will consume live sensor data from the registered Kafka topic.
- Stored sensor data can be fetched from Sensor Repository by providing Sensor ID to the Sensor Manager

ii. Scheduling on the platform :

1. Initial Application Deployment:
    - When the Platform UI(Developer) delivers the deployer.json to the scheduler through API Gateway the request with the scheduling information, the scheduler will either schedule a cron job or will put the deployment request in a time based queue.

2. End User Request:
    - Scheduler receives requests for scheduling new end user requests.
    - Uses a cron job scheduling which takes requests
    - Sends the request whose scheduled time has arrived to the Deployment Manager.

iii. Acceptance of scheduling configuration :

1. Initial Application Deployment:
    - The scheduling information provided by the Application Developer on the Platform UI generates deployer.json files automatically.
    - When the application is to be run, the Application Controller fetches application zip and hands it over to the Scheduler.
    - The Scheduler then scans the meta file and passes the meta data to the Deployer for deployment.

2. End User Request:
    - Application Controller receives end user requests for instantiation.
    - It then passes the instance information to Scheduler.

iv. Starting and Stopping services :

1. The scheduler checks the start time, end time of the service and gives it to the deployer.
2. The deployer communicates with the load balancer and node manager to create an instance of the service on the server node. The service can then be considered to have been started.
3. Once the end time of the service as provided by the scheduler is reached, the service is stopped, and the node is made free. The new node status is given to the node manager.
4. A normal service is added to the schedule queue only once, but a cron job service is added repeatedly.

v. Communication model :

1. The continuous stream of data produced by sensors and consumed by my platform(Sensor Manager)
2. Continuous monitoring of Modules by heartbeat monitor for fault tolerance
3. Inter module communication(eg: Communication between deployer and node manager)
    Synchronous Communication- HTTPs
    Asynchronous Communication – Messaging Queue(Kafka),HTTPs
4. Inter Containers/Nodes communication
    Sync Communication- HTTPs
    Async Communication – Messaging Queue(Kafka),HTTPs

vi. Server and service life cycle :

<u>Service Lifecycle:</u>

1. Consumer specifies the specifications and service policies.
2. The request for service is scheduled.
3. The scheduler hands service to the deployer.
4. The service is composed. It is packaged, bundled, and deployed on a node by the deployer. The node is provided by the node manager.
5. Once the end time of the service is reached, the scheduler sends the signal to the deployer to stop the particular service.

<u>Server Lifecycle:</u>

1. The node manager keeps information about the free and available nodes.
2. It provides a node on-demand whenever a request is made for the same by the deployer.
3. The request is made when the deployer needs to deploy a service.
4. A monitor keeps track of the node's status and restarts or reprovisions it if necessary.
5. The node is made free when the deployer gets the signal to stop a particular job running on the node and its status is updated by the node manager.

vii. Deployment of application on the platform :

1. The scheduling information provided by the Application Developer on the Platform UI generates deployer.json files automatically.
2. API gateway receives the request and forwards it to scheduler.
3. Scheduler sends a request to the deployer with the details of Application ID and Service ID to deploy.
4. Deployer upon receiving the request, downloads the application zip file from the blob storage and generates the Dockerfile.
5. If the docker image is not present in ACR then only the docker image is built and stored in Azure Container Repository(ACR) and its corresponding image path is stored in MongoDB with other details like Application ID and Service ID.
6. Deployer sends the request to the load balancer to give the best node.
7. After getting a response from the load balancer, which sends the username, password and ip address of the VM, image is fetched from ACR and run on the VM.
8. After running the image, the service is containerized in the VM.

viii. Registry & repository :

1. Application Registry-Application registry will contain the metadata of each application uploaded by the application developer.
2. Sensor Registry- The sensor registry will contain the metadata of each sensor registered by the application developer on the platform
3. Platform Repository-Platform Repository will contain the source code for Platform modules

ix. Load Balancing :

1. Load balancer plays a role between deployer and node manager.
2. Deployer will request a load balancer for the node to deploy a service. Load balancer will fetch the details of the nodes from the Node manager..

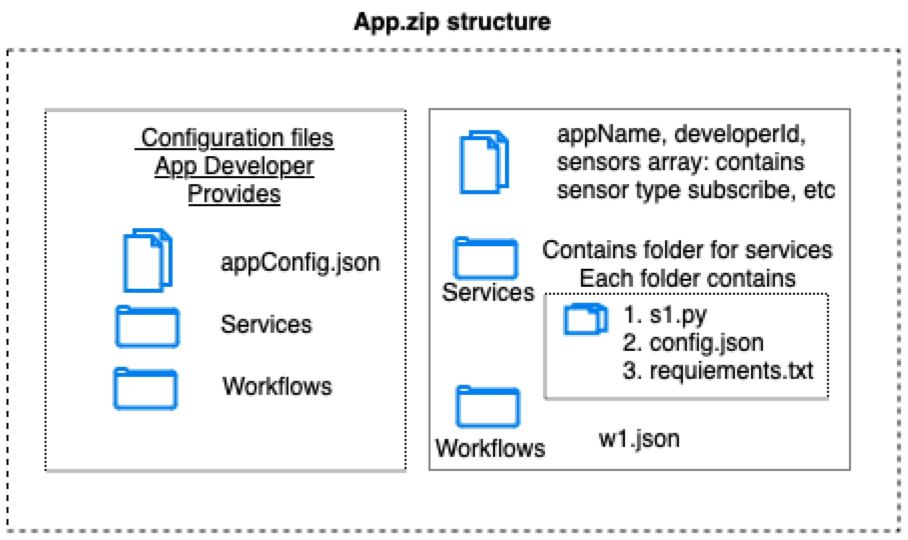
3. Load Balancer has a Load Balancing Algorithm that takes node details and gives a best node with lowest load. This best node details will be provided to the deployer.

x. Interactions between modules :

1. Boostrapper and Other Modules
   Sensor Manager, Application Manager and Deployer are initialised using this module.
2. Scheduler and Deployer
   Scheduler will reach the Deployer as per schedule. It will provide information about the application.
3. Application Conroller and Sensor Manager: Validator provides the sensor information to the Sensor manager (sensor id, sensor type, geolocation etc) upon sensor registration
4. Load Balancer and Deployer
   Deployer will request a load balancer for a node in order to run a service. Load Balancer will give details of the best node using the Load Balancing Algorithm.
5. Node Manager and Load Balancer
   Load Balancer contacts Node Manager for details of Nodes like number of services executing on nodes, type of services etc. Node Manager will provide these details to Load Balancer.
6. Monitoring & Fault Tolerance and Other Modules
   Monitoring module checks that other modules are working properly or not. If the module is down, it will take actions accordingly. It checks for instances of application manager, scheduler, deployer, load balancer, node manager. It also communicates with platform admin for information regarding failure in instance detection or reinitialization.

xi. Packaging details :

1. Application zip -


App.zip structure
Configuration files App Developer Provides
appConfig.json
Services
Workflows
appName, developerId, sensors array: contains sensor type subscribe, etc
Services
Contains folder for services Each folder contains
1. s1.py
2. config.json
3. requiements.txt
Workflows
w1.json

xii. Configuration files details :

1. Configuration files -
   - Appconfig.json - contains services data, workflows and sensor binding info(type, no of instances) per service
   - Config.json - present in each service
   - Workflow.json - Used to generate scripts to run workflow.

xiii. Interaction of different actors with the platform :

1. Application Developer:
   - This actor will deploy an application, developed by him, on our platform.
   - This actor will provide configuration files and source code in a zip file.
   - Application developers can monitor the status of their application on the platform dashboard.
2. Platform Admin:
   - He can monitor sensor data and device health on the dashboard.
   - He can create a new instance of a service instance if required.
3. Platform developer:
   - This actor is responsible for starting and stopping all major components in the platform.

4. User:
   - User will register the sensor in the application and the application will communicate with the application manager.
     They also start service on the platform through the application interface.

c. Non-Functional Requirements :

i. Fault tolerance :

It refers to the ability of a system to continue operating in the event of a failure or error.

1. Platform :
   Our platform will be able to handle failures in its components (such as the scheduler, the deployer, the application manager, etc.) without causing the entire platform to fail. This process will involve strategies such as redundancy (the ability to configure one or more backup components to take over for a component that fails) and automatic recovery. The platform will also detect failures and notify system administrators or other relevant parties regarding the same.

2. Application :
   Applications hosted on our platform should work, without crashing or losing data.
   In case of application failures, our platform would re-initialize another application instance.
   In case of node failures, our platform will reinitialize all the applications or containers running on the failed node to another node.

ii. Scalability :

It refers to the ability of a system to handle increasing amounts of work or users without degrading performance.

1. Platform :

Our platform will be able to handle an increasing number of applications, users, and data without impacting its performance or stability. This would involve strategies such as horizontal scaling (adding more resources such as servers or nodes) and using load balancing to monitor the performance of each node and scale up or down as needed.

2. Application :
   Our platform would have a monitoring system which checks if an application instance is consuming resources above a specified threshold. If such an instance is found, that instance will be shifted to a new node at runtime which exclusively belongs to this application instance.

iii. Accessibility of data :

1. Application :
   Application Developers will have to specify the type of all the sensors their applications will use in the configuration file and they will be able to access the data from the sensors through the API provided by our sensor manager.

2. Sensors :
   Sensors registered on our platform will retrieve and report data continuously. Based on the data retrieved certain events will be executed.

iv. UI and CLI for interaction :

1. Platform Admin:
   He can monitor sensor data and device health on the dashboard.
   He can create a new instance of a service instance if required.

2. Application Developer:
   He can upload an application zip file containing application code, required sensor binding information to the platform using UI.

3. User:
   Users can choose sensors from the application interface based on location and subscribe to them. Also, they can run the services provided by the application.

v. Security - Authentication and Authorization :

1. Authentication:
   Every user who would like to interact with the platform must be authenticated.
   All the users must be authenticated before using the platform.

2. Authorization:
   Different types of users have different privileges to access the platform.
   Platform admin has access to check the current status of the platform.
   Application Developer who deployed the application via our Platform can see the current status of platform and information of application which he deployed on platform.

vi. **Monitoring :**

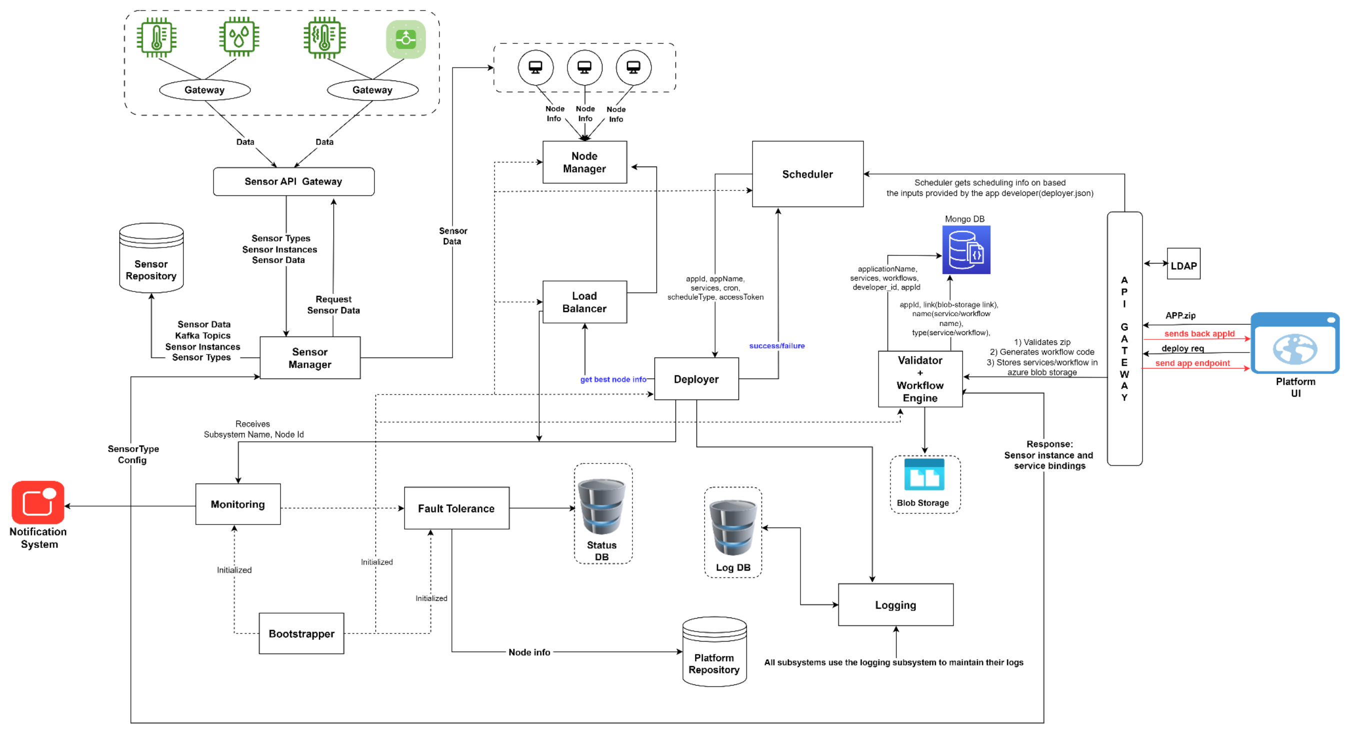
1. Platform admin and App Developer who deployed his app on platform can monitor the current status of the platform such as loads on the computation instances which they have, no. of requests coming from end users, current usage of bandwidth, memory and other resources of the platform, device health etc.
2. Platform admin can have access to the Monitoring web-view page where he can view the real time heartbeat of subsystems and check their status whether a subsystem is active(heartbeat rate normal) or alert mode (heartbeat has not come since the last 150 seconds).
3. This UI page contains an email notification tab where admin is provided with email broadcasting facilities to app deployers regarding any update of the Platform.

vii. **Persistence :**

1. The system can recover quickly from faults or errors, ensuring that it remains reliable and available.
2. When any module is found to be not working properly or down, the Fault tolerance module will take care. It will re-initialize that module.
3. Data for the state of modules will be in the relational database.
4. The fault tolerance module can then use this information to re-initialize the affected subsystem and restore its functionality.

4. List the key functions :

a. A block diagram listing all major components :


Gateway
Gateway
Data
Data
Sensor API Gateway
Node Info
Node Info
Node Info
Node Manager
Scheduler
Scheduler gets scheduling info on based the inputs provided by the app developer(deployer.json)
Sensor Repository
Sensor Types Sensor Instances Sensor Data
Request Sensor Data
Sensor Data
Sensor Data Kafka Topics Sensor Instances Sensor Types
Sensor Manager
Load Balancer
appId, appName, services, cron, scheduleType, accessToken
success/failure
get best node info
Deployer
Mongo DB
applicationName, services, workflows, developer_id, appId
appId, link(blob-storage link), name(service/workflow name), type(service/workflow),
1) Validates zip 2) Generates workflow code 3) Stores services/workflow in azure blob storage
Validator + Workflow Engine
A P I G A T E W A Y
LDAP
APP.zip
sends back appId
deploy req
send app endpoint
Platform UI
Receives Subsystem Name, Node Id
SensorType Config
Notification System
Monitoring
Fault Tolerance
Status DB
Log DB
Blob Storage
Response: Sensor instance and service bindings
Initialized
Initialized
Initialized
Bootstrapper
Node info
Platform Repository
Logging
All subsystems use the logging subsystem to maintain their logs

b. List the 5 major parts each team will take one part :

Team 1 - Deployer, Boostraper, LDAP, Platform UI
Team 2 - Application Controller(Validator & Workflow Manager), Scheduler, Kafka Central
Team 3 - Node Manager and Load Balancer
Team 4 - Sensor Manager
Team5 - Logging, Monitoring & Fault Tolerance

Use cases :

c. List what the users can do with the solution :

i. **Monitor sensor data:**
Users may see real-time sensor data such as temperature, humidity, pressure, and other sensor values in charts, graphs, and other data visualisations

ii. **Analyse sensor data:**
Users can examine sensor data to acquire insights into sensor activity, find patterns and trends, and identify abnormalities or issues that need to be addressed.

iii. **Configure sensor workflows:**
Users may define data pipelines, set up alerts and notifications, and automate data processing operations to establish processes for processing and analysing sensor data.

iv. **Develop and deploy custom applications:**
Within the platform, users may create and deploy bespoke apps such as dashboards, reporting, and integrations with other systems.

v. **Manage security and access:**
Using the platform, users may manage security and access by defining user roles and permissions, configuring multi-factor authentication, and encrypting data in transit and at rest.

vi. **Monitor system performance:**
Users may monitor platform performance by measuring system uptime and availability, as well as identifying and addressing issues that may influence system performance.

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

i. **End-users:**
End-users are those who utilise IoT devices that are linked to the platform. End users might be customers who use a smart thermostat or industrial personnel who utilise sensors to monitor machines. Their domain/vertical would be determined by the device or application they are utilising. They may not have a direct function in the platform, but they may require access to examine data, modify settings, or get notifications.

ii. **Administrators:**
Administrators are in charge of the platform's management, which includes establishing settings, managing users, and assuring system performance and security. They may work in a range of domains/verticals, such as information technology, operations, or engineering. Their responsibilities may include system administration, network administration, and application administration. They utilise the system to strive to maintain and optimise the platform so that it meets the demands of its consumers.

iii. **Developers:**
Developers are in charge of creating and implementing bespoke applications on the platform, such as dashboards, analytics tools, and system integrations. They might work in a range of domains/verticals, including software development, data science, and engineering. They may work as a full-stack developer, front-end developer, or data engineer. They utilise the system to try to add new capabilities or enhance current functionality in order to fulfil the demands of end-users or administrators.

iv. **Business users:**
Finance, marketing, and operations are some of the domains/verticals in which business users may work. Companies might utilise the platform to obtain insights into data that would help them make business choices, such as streamlining supply chain operations or recognising market trends. Their responsibilities might range from business analyst to operations manager to marketing manager. They utilise the technology to obtain insights into data that might help them make business decisions.

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

i. **Farm management System App** :
To increase the productivity of agricultural and farming processes in order to improve yields and cost-effectiveness with sensor data collection by sensors like soil moisture, temperature sensor, ph sensor, water level sensor. We will use sensor data collection for measurements to make accurate decisions in future. It can be used as a reference for members of the agricultural industry to improve and develop the use of IoT to enhance agricultural production efficiencies. The sensor types are air-condition-sensor, for measuring the temperature of the soil, and soil-moisture-sensor, for measuring the humidity of the sensor and turning off or on the sprinkler based on the threshold.

ii. **Smart fire detection and Sprinkler System** :
The smart fire detection system uses various sensors such as smoke detectors, heat detectors, and flame detectors to detect the presence of fire. These sensors are connected to a central control panel that can identify the location of the fire and alert the building occupants and the fire department.The sprinkler system is designed to automatically release water or other extinguishing agents to extinguish the fire once it has been detected. The sprinkler system is typically composed of a network of pipes, sprinkler heads, and control valves.In a smart fire detection and sprinkler system, the sprinkler system can be activated selectively based on the location of the fire, thanks to the information provided by the smart fire detection system. This allows for more efficient use of water and other resources, as well as reducing water damage to unaffected areas of the building.

iii. **Waste Management** :
Waste management is an important aspect of modern cities, and the Internet of Things (IoT) can be leveraged to create smart waste management solutions that optimise waste collection, disposal, and recycling processes. The sensors that will be used are :
Ultrasonic sensors: These sensors are used to measure the fill-level of waste in containers, such as trash bins or dumpsters.

Weight sensors: These sensors are used to measure the weight of waste in containers, such as garbage trucks or compactors.
GPS sensors: These sensors are used to track the location of waste management equipment, such as garbage trucks or recycling vehicles.
Temperature sensors: These sensors can be used to monitor the temperature of waste in landfills or composting sites.
Some of the services we can provide are : Real-time monitoring, Predictive maintenance, Smart routing, Recycling optimization, Cost reduction.

iv. **Smart Home** :
Managing home appliances, theft detection using automation of all its embedded technology. It defines a residence that has appliances, lighting, heating, air conditioning, TVs, computers, entertainment systems, big home appliances such as washers/dryers and refrigerators/freezers, security and camera systems capable of communicating with each other and being controlled remotely by a time schedule, phone, mobile or internet. All sensors are connected to a central hub controlled by the user using a wall-mounted terminal or mobile unit connected to internet cloud services.

v. **Traffic Monitoring** :
Traffic monitoring is an important application of the Internet of Things (IoT) that can help cities and transportation agencies optimise traffic flow, reduce congestion, and improve safety. The sensors that will be used are :
Inductive loop sensors: These sensors are embedded in the roadway and detect the presence of vehicles by measuring changes in the electromagnetic field as the vehicle passes over the sensor.
Infrared sensors: These sensors use infrared beams to detect the presence of vehicles and measure traffic volume, speed, and occupancy.
Video cameras: Video cameras can be used to monitor traffic conditions and detect accidents or other incidents.
Radar sensors: These sensors use radio waves to detect the presence of vehicles and measure traffic speed and volume.
Ultrasonic sensors: These sensors are used to measure the distance between vehicles and can be used to detect and prevent rear-end collisions.
Some of the services we can provide are : Real-time data collection, Smart traffic management, Predictive maintenance, Intelligent parking, Emergency response.

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

a. Name of use case :

Farmer irrigation

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

Agriculture

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

The purpose of this use case is to provide an automated irrigation system for farmers that will optimise water usage and improve crop yields. The irrigation system will monitor irrigation time, soil moisture levels and weather conditions.
The system will interact with farmers through a user interface, allowing them to view real-time data (air quality, moisture, successful season crops growth). This data is fetched from the sensor database via the sensor manager. It will also allow users to view previous years crops like selling price profit, most selling crops and will also allow users to set certain event based workflows such as start sprinkler actuator at a particular time. This benefits the farmers to grow high yield crops, reduced water usage, and improved efficiency.

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

Collect and analyse large amounts of data from multiple sensors such as soil moisture sensors, air quality sensors, etc and use their data to perform weather forecasts, suggest crop growth models (seasonal crops). This data can also be used to optimise water usage and improve crop yields.

e. Information model :

The information model for an irrigation system includes data on soil moisture levels, weather conditions, crop growth, and water usage. This information is collected and processed by the system to determine the optimal seasonal crops to grow and optimise irrigation schedule and amount of water to apply.

f. How is location and sensory information used :

Location information can be used to provide farmers with site-specific season wise crop recommendations based on soil type, and other factors. Sensory information such as soil moisture sensors and weather stations can be used to collect real-time data on environmental conditions and adjust irrigation schedules and methodology accordingly.

g. Processing logic on the backend :

Collecting, preprocessing and analysing data from the sensor database. This includes integrating various sensor data, weather API’s to make real-time decisions like which crop to grow, how much water to apply, etc.

h. User’s UI view- what is their UI, where and all will they do with these systems :

Users UI View will contain real time data on soil moisture level, weather conditions and irrigation schedules (weekly / monthly schedules). Farmers can adjust settings such as setting thresholds for various events, adjust irrigation duration and timings, etc. The interface can be accessed through a mobile and desktop, allowing farmers to control and monitor the system from anywhere.