Design Requirements
For
IoT Based Application Platform

Internals Of Application Server

Submitted By : Group 6 Team 3

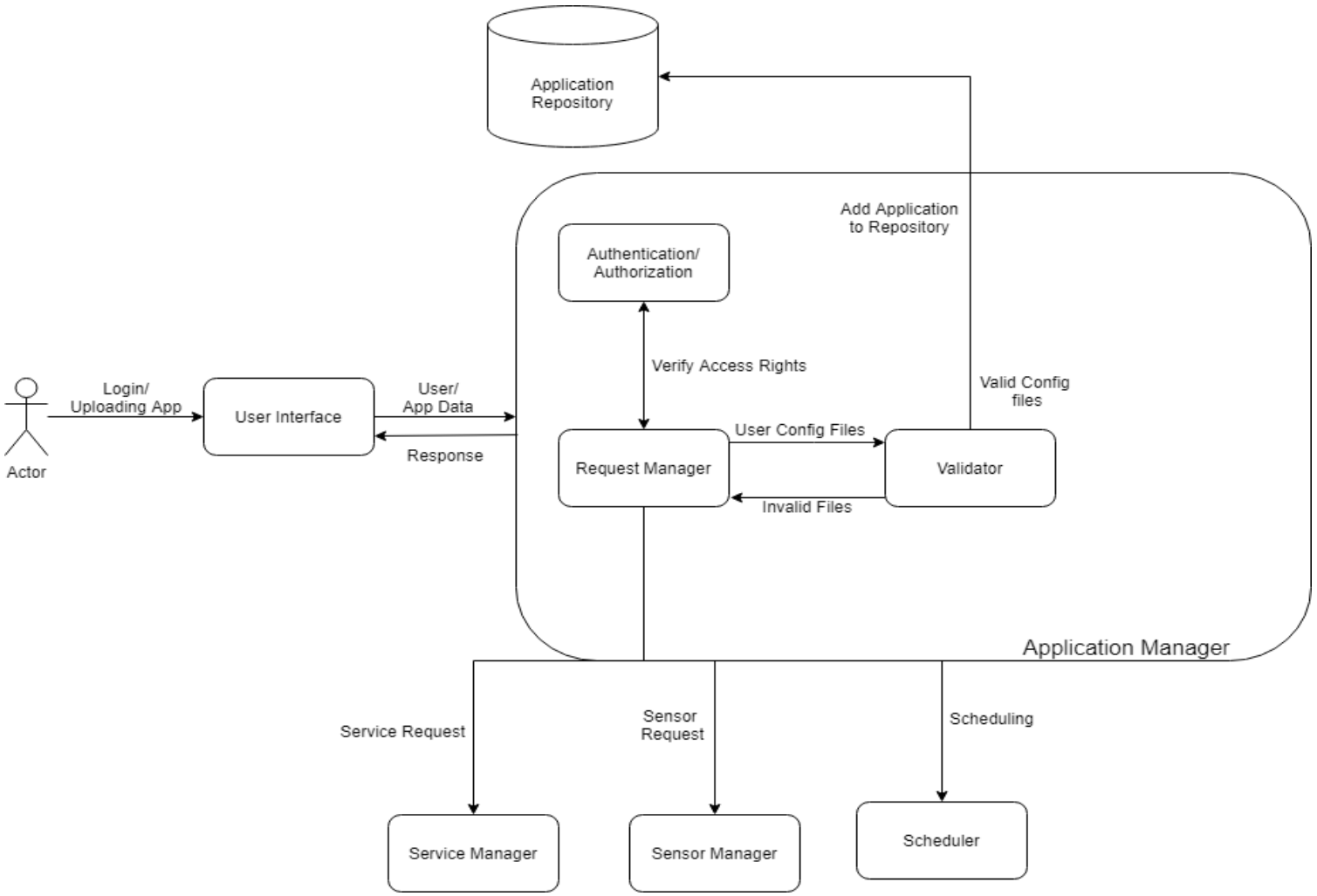
Abhijeet Srivastava(2020202011)

Shubham Singh(2020202002)

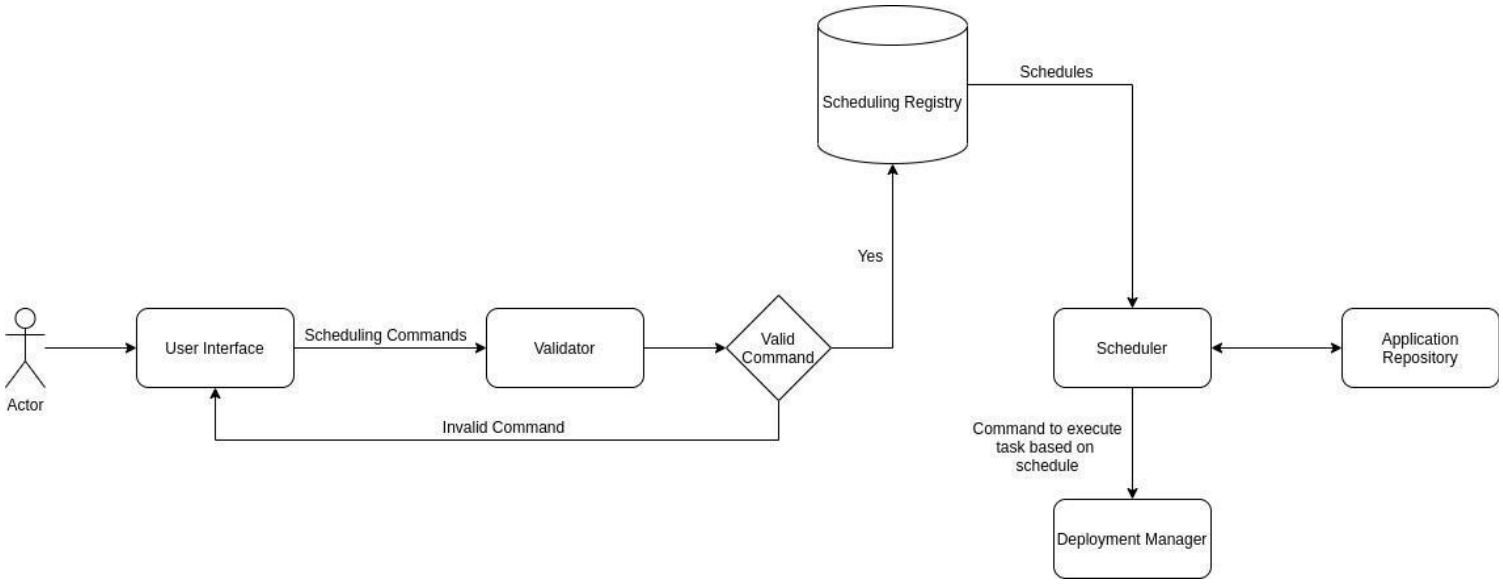
1. **Introduction to the modules**-

Application Manager will process incoming requests from users and will validate if the requests are sent in the specified format. The requests include uploading and deploying the application. If the requests are valid, they are forwarded to the appropriate components.

Scheduler will read the schedules from the registry and send the commands to the deployment manager to execute the algorithm based on the schedule.



Block Diagram of Application Manager



Block Diagram of Scheduler

2. **Use Cases** -

- **Request Handling** - The application manager will accept and authenticate requests.
- **Web GUI** - Web GUI Dashboard will provide an interface to the users based on their access rights.
- **File format Validation** - The uploaded config files will be validated if they are in specified format.
- **Scheduling -** The user will specify schedules for algorithms.

3. **Test Cases** -

- **Check User Registration -** Check if the user is registered or not .
  Input - Username and Password
  Output - Returns dashboard if the user is registered or Sign up Page for registration

- **Check User Role-** Check the access rights of the user.
  Input - Username and Password and Role
  Output - Returns the services accessible by the user role.

- **Check scheduling -** Check the scheduling type for Now or Scheduled execution. If the scheduling type is scheduled then service is scheduled for start time and end time. If the scheduling type is now then the service is scheduled for now.

  Input - start time - hh:mm
  - end time - hh:mm
  Output - Return boolean whether the time is valid or not.

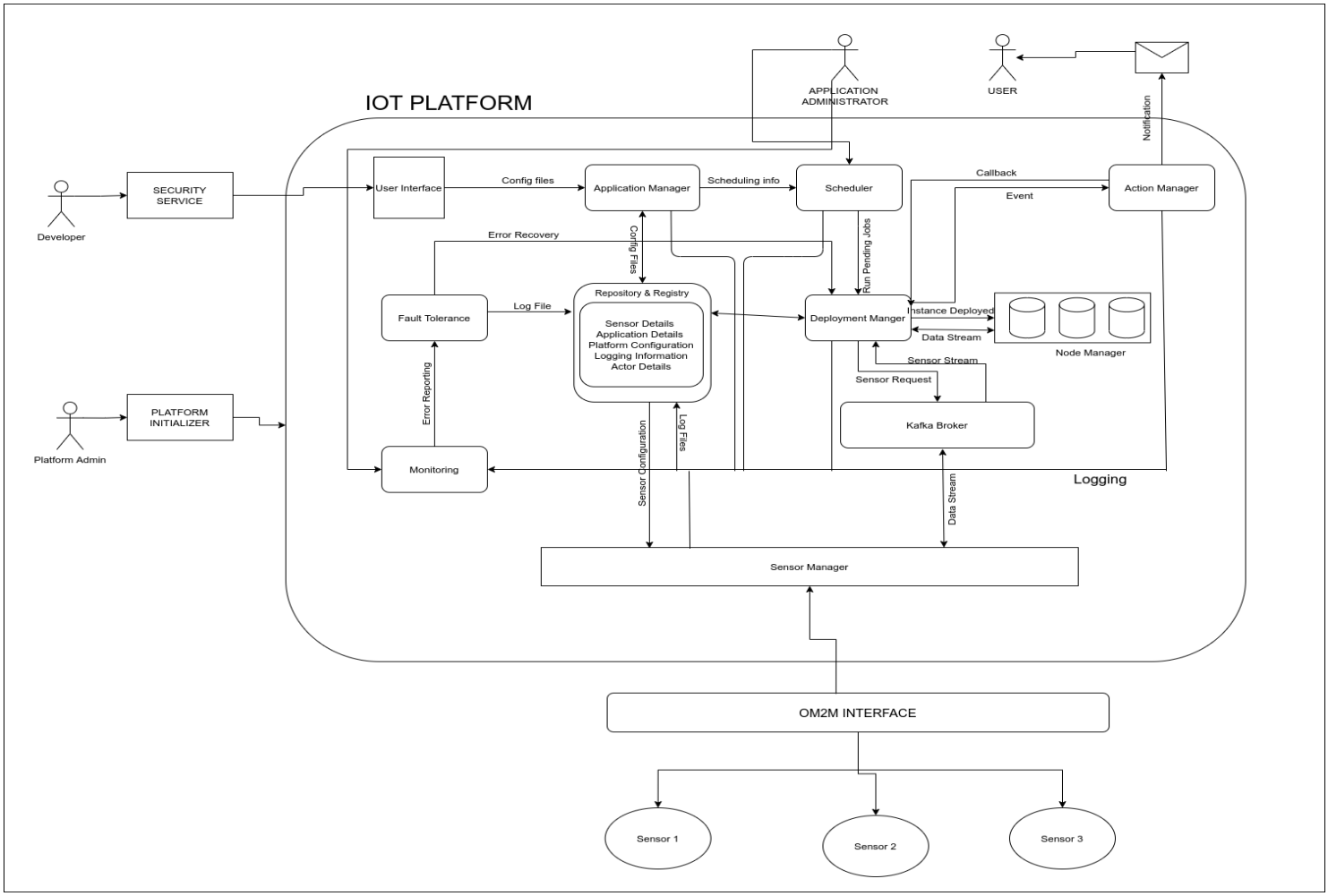
- **Verify file Integrity -** Check if the archive is valid and contains all the files required.
  Input - Zip archive
  Output - Boolean value indicating the valid archive

- **Verify file formats -** Check the file formats
  Input - Files
  Output - Boolean value indicating valid and invalid file formats.

- **Stop currently running algorithm -** Stop a particular service (currently executing) created by the user.
  Input - Stop command by user
  Output - Execution status of service changes to stopped.

4. Solution Design Considerations -

4.1 Design Big picture



4.2 Environments to be used

- 64-bit OS(Linux)
- Minimum RAM requirements: 4GB
- Processor: Intel i3 5th Generation

4.3 Technologies to be used

Flask

Why: Tool provided by it will help to expose Api endpoints for the platform to use and it gives developers a good base framework of request/response management to work with.

Bootstrap
Why: it will be used for UI to build responsive web applications.

Kafka
Why: it will be used for managing communications among various modules

MongoDB
Why: For use as repository and Database

Python
Why: for deploying and developing platform

Schedule

Why: for schedule processing

4.4 Overall System flow and Interaction

We will initiate the bootstrap or platform initializer program to initialize our platform. The bootstrap module will make sure that our platform will be up and running.

It will then start the additional external services like Kafka and the database for the repository.

After all the components are running, we will initiate the web interface.

The application manager will start the web interface and accept the request to authenticate it and then will ask for the zipped files.

After the zipped files are uploaded the application manager will extract the zipped files and validate them.

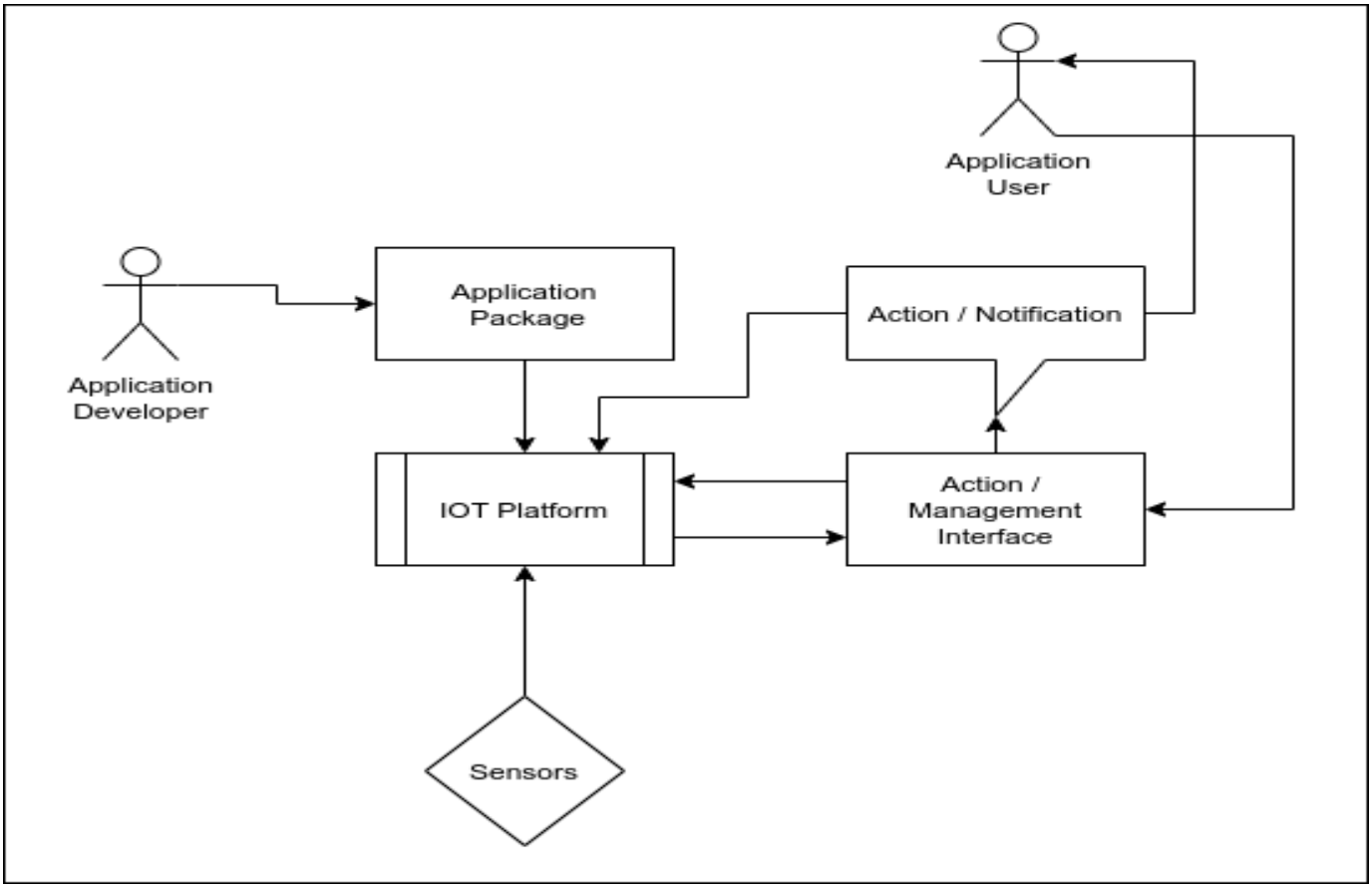
The scheduler will read the application configuration file and create a schedule instance. The scheduled instance will have an algoid , start_time , end_time ,job_type and then it will execute accordingly.

Once the scheduler triggers the deployment module at the time a particular algorithm must be run, the deployer will deploy the application, gives some status, and deploys the application.

The algorithm running can also trigger some actions based on certain events and these actions are taken care of by the action Module.

The monitoring modules monitor all the platform components for their status and if some components go down then restores them to their running state.

4.5 IOT System Design:



The application developer will develop an algorithm or application which based on user requirements performs some operations on the data provided by the sensors and will send the result back to the users in required response format I.e. either a notification or some action on the sensor itself.

4.6 Approach for communication & connectivity:

We are using Kafka to enable communication between different components of the platform. Initially the different platform components will be registered with the Kafka broker and will be sending status/messages/acknowledgements in predefined format.

4.7 Registry & repository:

The Repository will be used to store different files related to different modules and will be separated based on the modules so that each module can access its repository and use them as required.

- **Application Repository** : To store application Configuration and executables.
- **Credentials Repository** : To store credentials of different actors interacting with the platform.
- **Scheduling Repository :** To store scheduling details of the applications.

4.8 Scheduler:

The scheduler takes the input from the user on when the service should be deployed and some other details like if that job is a recurring job and when to stop it. It creates and sends a request to the deployer when that trigger point has been reached and gets the status returned by the deployer.

4.9 Interaction with other Components (Internal design overview)

★ **Platform Initializer and Others**
Platform Initializer passes configuration requirements to all the platform services and deploys them on servers after fetching them from the platform repository.

★ **Application manager and Scheduler**
The Application manager will send application service name that needs to be scheduled which will be forwarded to the deployment manager according to the schedule given by the user.

★ **Scheduler and Deployment Manager**
Scheduler passes the application and scheduling details to the deployment manager which uses it to deploy applications on servers.

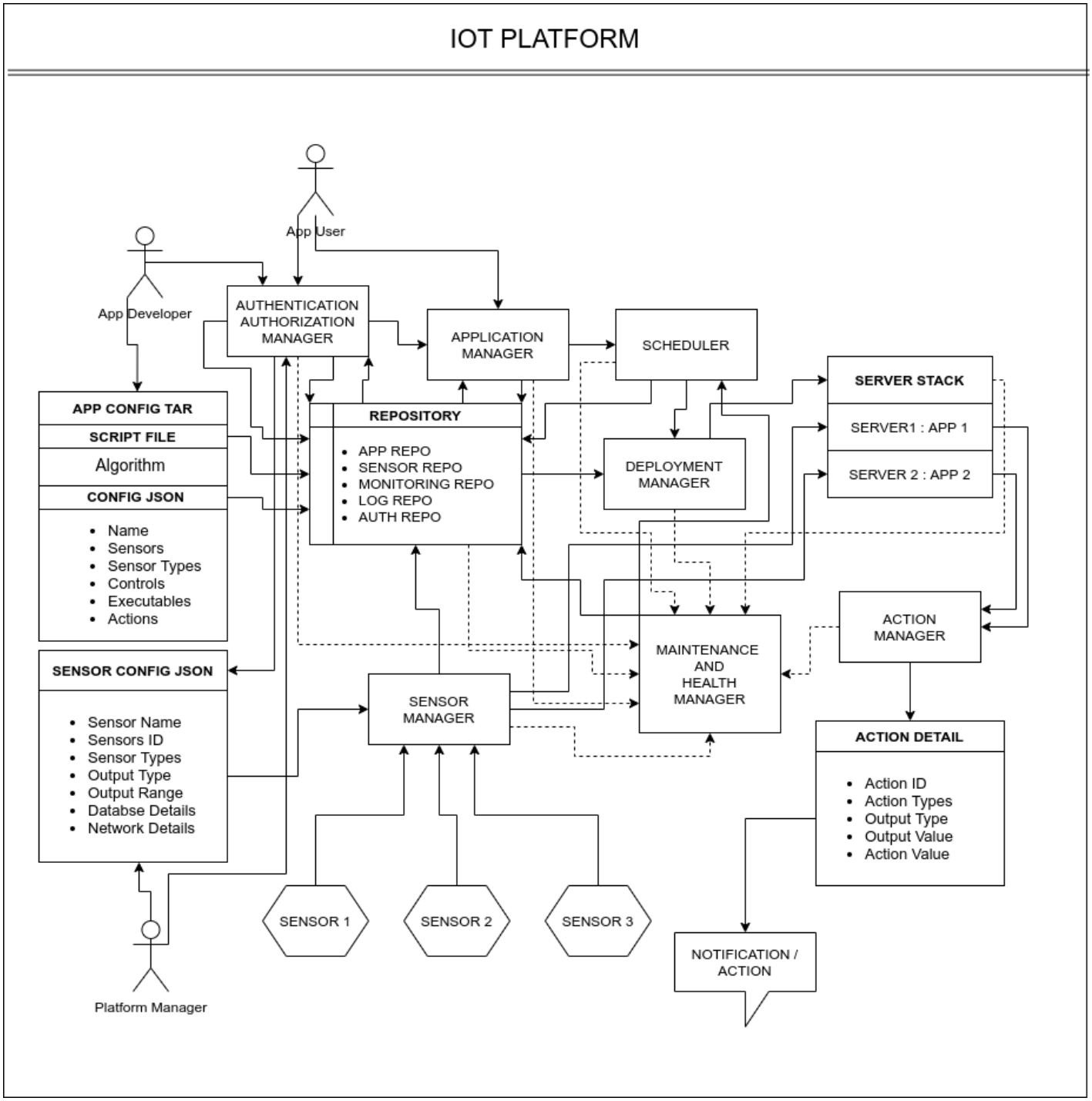
★ **Application Manager to Sensor Manager**
Application Manager sends the details of sensors registered by the platform administrator to sensor manager.

★ **Monitoring and Fault Tolerance and Others**
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.

4.10 Wire and file formats

- The main package files are provided in the zipped manner.
- The zipped files consist of the application, application configuration file, sensor configuration file.
- The application file is a python script.
- The application configuration file is json.
- The sensor configuration file is json.

5. Application Model and User's view of system -



Key elements

1. **Application Config file** - Contains info about the application used i.e., type of sensor used, name of sensor, controls, executables - all the scripts used in application.
2. **Sensor Config file -** contains details about the sensor used by the platform like sensor name, id, type, output type, output range, network details etc.
3. **Application Platform -** Sends config file to the platform and receives output from the platform.
4. **Sensor Registration -** if the application needs new sensors, then it will request the platform to add it, and the sensor registration module will store all information about the new sensor in the database and register it.
5. **Platform manager -** Manages the flow of the application and when a new application is received, provides it with resources.

Various Users

1. **Application developer** - develops applications, its algorithms and what all applications are going to do.
2. **Platform Admin** - Admin registers the sensors on platform.
3. **User -** Users who will be using the application.

Key Steps in whole application development Process -

Develop:

- Understanding the platform and develop application based on a Config file containing -
    - Application Name
    - Application Dependencies
    - List of Algorithms in Applications
        - Algorithm Name
        - Algorithm Id
        - Type and number of Sensors used by algorithm
        - Action performed by algorithm

Algorithm Script -

- Contains info about all the sensors to be used based on type.
- What elements are to be controlled.
- What type format of data is needed.
- Processing and interference from the data received.
- Notification to be sent.

- First a <app_name>.zip file must be provided by application developers which contains two pieces of information like app_config.json and scripting files. The app_config.json contains information mentioned above and scripting files contain actual algorithms.
- The developed application must be compatible with our platform.

Deploy and configure:

- Now these details can be used and our sensors should be registered and then details like the running of sensors is given by the developer according to which our sensor will run and now when the app dev wants any sensor data then he/she will request for that .
- And then sensor manager will provide the corresponding data from that sensor sensor manager and sensor manager will pass on data to the deployer which runs that on our algorithm and sends the output to the action handler, which will further process the output in a way that depends on sensor type.

Run and monitor:

- Contain info like when to run a particular sensor or whether it will run always or on schedule and monitoring of the system.
- Application will be executed on a docker instance with the dependencies running.
- Output of an algo instance, if it is not recurring, can be seen on going to the particular algorithm instance's page.
- Monitoring will be done on platform services and in case of a services failure the application will handle it with the fault tolerant system.

6. Key Data Structures -

Sensor Information Structure

- Sensor ID
- Sensor Type
- Location
- Sensor Input Data Type
- Sensor Data Rate

Scheduling Information

- Schedule Type
- Schedule Algo
- Schedule Start Time
- Schedule Stop Time

User Information

- User Role
- User Email
- User Password

Algorithm Information

- Algorithm Name
- Algorithm Id
- Algorithm Sensors
- Algorithm Action Type

Registries and Repositories -

- Scheduling Registry - To store all schedules provided by the user.
- Application Repository - To store all the applications/algorithms.

7. Interactions and Interfaces -

User Interactions - The user can login using a web dashboard using ID and password. The user can upload, deploy and manage the application using the dashboard after successful authorization.

File Formats -
Application Package File - zip file
Application Config File Format - JSON File
Sensor Config File Format - JSON File
Python Files/Algorithms - py File

Interaction with other modules -

1. Scheduler will send the details of the algorithm with the deployer which has to be executed. Apart from algorithm/application it will also share details regarding the dependencies that have to be installed before the request has to be executed. Deployer will also share the id of the algorithm so that if it decides to kill the job midway it can do so.

2. Application Manager will interact with scheduler's registry to store schedules and with deployment manager to get status of algorithms.

3. Application Manager will interact with sensor manager to register the sensors given from the user interface.

8. Persistence -

- **User Information -** The user registration is stored in the registry.
- **Scheduling Information -** The scheduling information is also stored in the scheduling registry.
- **Application Repository -** Store the application registry information.