REPORT
(Group no. 5 - Team 2)

Hackathon Video Submission -

https://iiitaphyd-my.sharepoint.com/:v:/g/personal/paras_joshi_students_iiit_ac_in/ETAn5iPlzM1FgUST4XlzVQkB0ASQh77aJCoaF2k94j9TeQ?e=v21q63

Application Workflow:

1) Different types of users (platform application developers, end users) can access the platform.

2) The Application developer needs to register themselves using suitable user id and password.

3) After successful registration, the Application developer needs to provide the name of the script file (in zip format) containing algorithm file (in .py format) containing the logic for the use case supported by the application on the platform and a config file (in json format).

4) The Application Manager module in the platform will then extract this zip file to retrieve the algorithm file and the config file and will then validate the config file to check whether the requirements mentioned are in suitable format like whether the sensor types are supported by the platform or not or whether the output file format is valid or not etc.

5) Different algorithms corresponding to different use cases are mapped using a list.

6) Different classes for different sensor types are maintained in a separate file.

7) When any end user wants to access a use case of the application like in this case, we need to find the average temperature of the body, the user first registers himself and then he needs to enter the application use case on the terminal.

8) After the request for the use case is submitted by the user, the algorithm corresponding to the entered use case is mapped and the algorithm is triggered

and the output is generated and printed on the terminal or in the file as mentioned in the config file.

Different functions work --

1.) **App_manager.py file -**---

a.) It is an application server that handles as the core for application deployment for various applications deployed.

b.) It contains various functions as --

Signup -- handles the signup feature of the application.

Login -- handles the login of the various user and application user.

c.) *Validates* the config.json file uploaded by the application developer.

d.) *Creates sensor instances* of sensors which the application developer wants to register on the platform.

e.) interacts with the *sensor manager* to get the sensor_output files when the end user requests.

2.) **deployer_user.py file** ---

a.) Different types of users that communicate with application manager module via this file include application developers and end users.

b.) The application developers send a registration request to the Application manager module and after successful registration, the application developer can then share a script.zip file containing files for algorithms and configuration files for different use cases supported by the application.

c.) The end users can enter the use case which they want to access via terminal prompt and the application manager then invokes the necessary algorithm required to process the requested use case and the result is returned via terminal or output file.

3.) **Sensorfile.py** ---

a.) Contains different sensor types that the platform supports as classes.

b.) It acts as the sensor manager which binds different types of sensors that the platform supports and generates a data flow at a particular instant of time when the request is made.

c.) The application manager interacts with the sensor manager if it wants to create a new sensor instance which then starts generating the data.

d.) The different sensors connected to the sensor manager continuously generate and output the data into the file named as the unique id given to each of the sensors, i.e the sensor id.

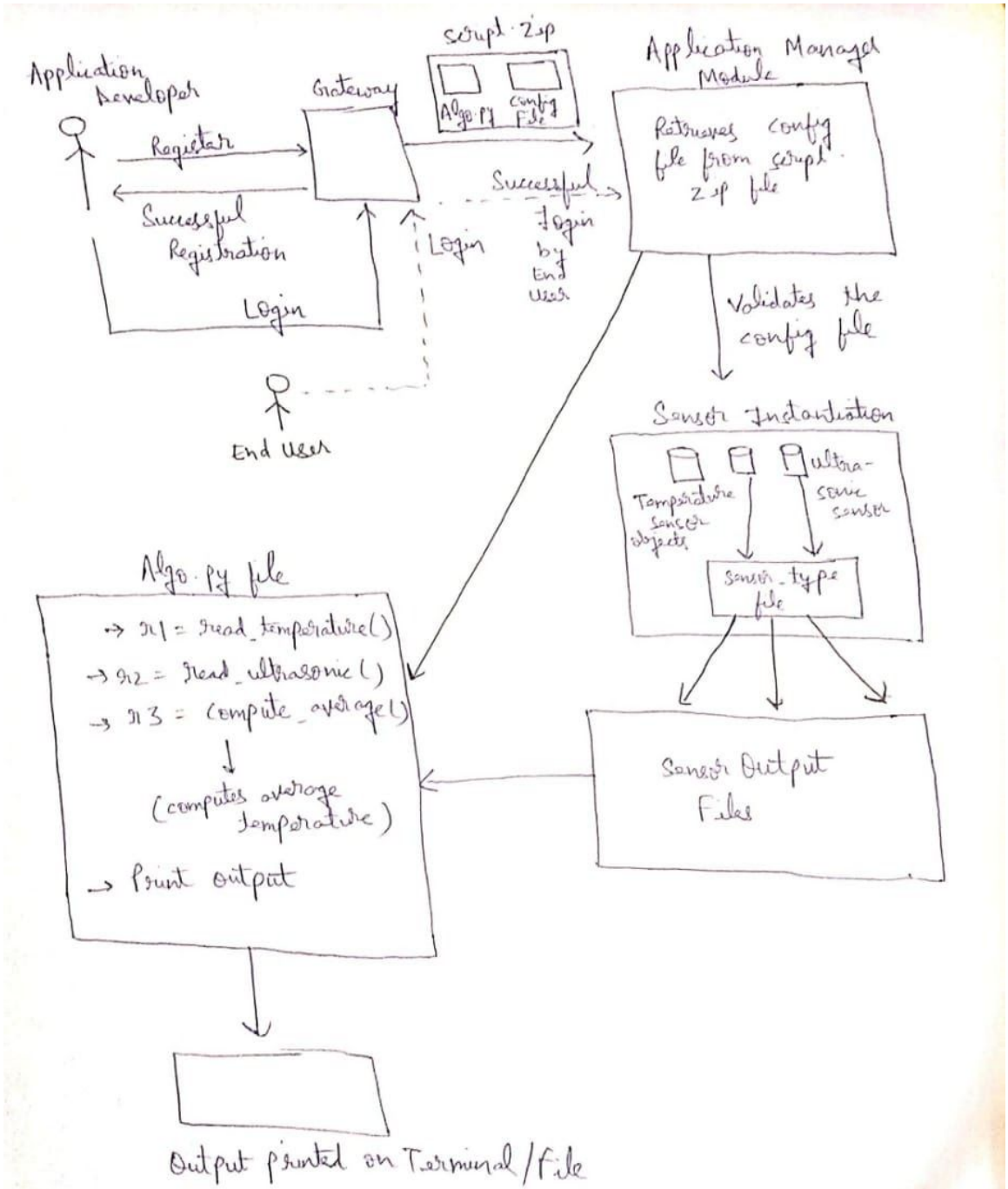
4.) Algo.py --

a.) It is uploaded by the application developer along with the zip file.

b.) It contains the methods and control flow of the algorithm to handle various sensor instances data and their basic data getting functions from the application platform and use them accordingly as per needs.

c.) This file is being run with the proper use-case by the end user and he/she will get the output as mentioned by the algorithm in this file.

Workflow Diagram:


Application Developer
Gateway
script.zip
Algo.py
config File
Application Manager Module
Register
Successful Registration
Login
Successful login by End User
Retrieves config file from script.zip file
Validates the config file
End User
Sensor Instantiation
Temperature sensor objects
ultra-sonic sensor
sensor_type file
Algo.py file
→ x1 = read_temperature()
→ x2 = read_ultrasonic()
→ x3 = compute_average()
(computes average temperature)
→ Print output
Sensor Output Files
Output printed on Terminal/File