

IAS Group 7 : Team 2

Team Design Document

Param Pujara : 2020202008

Nisarg Pethani : 2020202012

Karan Bhut : 2020202015

Contents

1	Functional Overview	3
2	Test Cases	4
2.1	Validator	4
2.2	Sensor data binding	4
2.3	Application manager	4
3	Solution design considerations	5
3.1	Environment to be used	5
3.2	Technologies to be used	5
3.3	Approach for communication & connectivity	5
3.4	Registry & repository on Azure	5
4	Low Level Design	6
4.1	Life cycle of the Module:	6
4.2	List of submodules	6
4.3	Brief overview of each sub module	6
4.3.1	Application Manager:	6
4.3.2	Sensor Data binding	7
4.3.3	Notification Manager	8
4.3.4	User interface	8
4.3.5	Azure MySQL Handling	8
4.3.6	Azure Repository	8
4.3.7	Application	9
4.4	Interactions between outer modules	10
4.4.1	Application manager with Scheduler	10
4.4.2	Application manager with Monitoring and Fault tolerance	10
4.4.3	Application manager with Sensor Manager	10
4.4.4	Application manager with Controller Manager	10
4.5	Interactions between sub-modules	10
4.5.1	User-Interface with Authentication module	10
4.5.2	User-Interface with Validator	11
4.5.3	Application manager and Sensor data binder	11
4.5.4	User-Interface and notification manager	11
5	Output	12

1 Functional Overview

Our module will take configuration files from Application developers and Platform Configurers through UI. We will validate structure of zip files through validation module. We will provide Authentication through UI. Users can login and deploy application they wanted, they can also see the notifications for application. We are also maintaining Azure MySQL database and Azure Repository to store whole project's data in cloud. We have also built Sample Application 1 which provides transport facility using buses.

2 Test Cases

2.1 Validator

- Test case 1:
 - Input: Zip file
 - Output: Valid or invalid
 - Description: It will check all the necessary file is present or not. Checking the Files contains all required information or not. Zip could be uploaded by:
 1. Application zip uploaded by application developer
 2. Sensor type configuration zip by sensor type configurer
 3. Sensor instance configuration zip by sensor instance configurer

2.2 Sensor data binding

- Test case 2:
 - Input: information to deploy application (request by application deployer)
 - Output: Send scheduling info and other necessary info to scheduler
 - Description: This module will receive the request of application deployer for application deployment. Request will contain the area where application sensors resides. So, application manager will find the sensor id of requested area. And binding will be done by application manager

2.3 Application manager

- Test case 3 (Notification Manager):
 - Input: application generates the notification which will be sent application manager
 - Output: It will redirect that notification to user's user interface using necessary API.
- Test case 4 (Authentication Module):
 - Input: User's username and id
 - Output: Successful login or Not.
 - Description : Login can be done by Application developer, End User or Sensor Instance configurer.
- Test case 5 (Notification Manager):
 - Input: User's username and id
 - Output: User specific notifications
 - Description : Notifications contain the specific application id for which notification is released.

3 Solution design considerations

3.1 Environment to be used

- OS: Linux

3.2 Technologies to be used

- Python
- Kafka
- Docker
- Why?
 - Improved Productivity
 - Interpreted Language
 - Dynamically Typed
 - Vast Libraries Support
 - Portability

3.3 Approach for communication & connectivity

- Kafka as Message Queue
- Publisher and Subscriber Model

3.4 Registry & repository on Azure

- Platform Database
- User Database
- Application Database
- Monitoring Database

4 Low Level Design

4.1 Life cycle of the Module:

1. Application developer and user will have to register first and then they can login on platform. Platform configurer will have to enter a secret key, after applying which he can register and login on platform.
2. Azure MySQL is used for storing different actor's credentials.
3. Platform configurer will upload different configuration files like sensor-type, sensor-instance, controller-type and controller-instance to the platform.
4. Application developer will upload application zip file containing code files and required sensors-controllers information.
5. User will provide information like place, date and time of deployment.
6. All these information will be stored in Azure Repository. Scheduler will take application information and sends it to nodes for execution.
7. During execution, user can go to the notification page of UI and see notifications coming from application.

4.2 List of submodules

1. Application Manager
 - Authentication module
 - Validation module
2. Notification Manager
3. Sensor data binding
4. User interface
5. Azure Database Handling
6. Azure Repository
7. Application

4.3 Brief overview of each sub module

4.3.1 Application Manager:

- Authentication Module
 - Authenticates Application developers and Users through their passwords. It also authenticates Platform Configurer with his password and Secret Key.
- Validation
 - Validates structure of configuration zips submitted by platform configurer and application developer.
 - When actors submit files in valid format. Below example shows sensor type upload page.

Home

Sensor Type Upload

Select zip file containing sensor-types

Browse... No file selected.

Submit

Validation successful: All validation has been done + All things has been saved in database

Need to upload more components? [click here to go back to Platform configurer home page](#)

- Error messages will be shown to actors just like above image. Instead of Successful message, error message will be displayed. Some of the error messages are given below.

1. when platform configurer tries to upload Zip file having incorrect structure:

Validation Failed: Wrong Directory Structue OR Unnecessary file/files have found in zip

2. when application developer tries to submit application whose required sensors are not present in platform.

Validation Failed: Invalid Sensor type... Sensor Type: ['sensor_type_2'] already exist

4.3.2 Sensor Data binding

- This module will bind sensor instance with index mentioned in application zip.
- User will provide location information of sensor instance and application developer would have mentioned Sensor type information in the App.
- This module will search a sensor info and sensor type info into database and select sensors from query output.
- Selected sensor will be removed from query result for that particular app.
- So, Same sensor will not be selected again in same application.
- Sensor data binder will binds all sensors of an application in this manner.
- It will reply success message or error if any happens to be occurred.
- The same binding process will be followed for controller data binding by this module.
- This is how sensor and controller data binding happens.

4.3.3 Notification Manager

- It will display notifications of different Applications executed by users.

4.3.4 User interface

- It includes Registration and Login pages for platform configurer, application developer and users.
- It also includes facility for platform configurer to upload different ZIP files like sensor_type, sensor_instance, controller_type and controller_instance.
- Application developer can upload application.zip file containing application requirements and code.
- user can deploy Application through it. User can also see different notifications of applications previously deployed.

4.3.5 Azure MySQL Handling

```
mysql> show tables;
+-----+
| Tables_in_hackathon |
+-----+
| app_1bac76ac43694b2299a66f980872dbf1 |
| apps |
| configurers |
| controllerinstance |
| controllerinstanceipport |
| controllertypes |
| deploy |
| developers |
| nodes |
| sensorinstance |
| sensorinstanceipport |
| sensortypes |
| users |
+-----+
```

Figure 1: Azure MySQL database tables

- Above image shows different tables present in MySQL database, We have managed above database for whole Project.
- It includes tables to store data of different sub-systems like different sensors information, application binding information, nodes information, etc.

4.3.6 Azure Repository

- We have managed Azure Repository for entire project to store codes for different module, configuration files for Kafka and Docke and json files for sensors and controllers. So that we can access them from distributed locations remotely.

Home > karan >

ias ...
File share

» [Upload](#) [Add directory](#) [Refresh](#) [Delete directory](#) [Properties](#)

Name	Type
[-]	
api	Directory
apps_info	Directory
controller_instance_info	Directory
controller_type_info	Directory
sensor_instance_info	Directory
sensor_type_info	Directory

Figure 2: Azure Repository

4.3.7 Application

- Sample Application 1
 - Application is built to provide transport facility using buses.
 - Sensors:
 - * GPS: Sends placeholder-id co-ordinates. Each bus is installed with one such sensor, one sensor is at IIIT-H campus and each police barricade has one such sensor.
 - * Biometric: Sends placeholder-id person-id (it will be unique on place-id level)
 - * Temperature: Sends current temperature of the place
 - * Light Sensors: Sends Lux Level of the place
 - Controllers: Lights, ACs, Buzzers
 - Cases:
 - * Whenever someone boards a bus, he/she will do biometric check-in. Fare will be calculated based on distance multiplied by fixed rate that information(Bus-id, Passanger-id and Fare) will be sent to guard.
 - * If temperature is more than a threshold switch on the air conditions or lighting is lower than a lux level switch on the lights.
 - * If more than two (≥ 3) buses come in a circle of given radius, except one send buzzer command to the rest. Run this service after every 1 minute.
 - * Also, Whenever a bus comes closer to a barricade by a threshold distance start/trigger an algorithm which sends an email to administration mentioning unique identifier of the bus an email body containing information of bus and notifying them.

Sent from your Twilio trial account -
[2021-05-08 17:18:19.774386](#):
Person (PersonID:
PersonID_[202050770](#)) is boarding
at Location ([1.15](#), [2.6](#)) in BusID: 4
Fair: 56.[859475903318](#)

Figure 3: SMS from Application

4.4 Interactions between outer modules

4.4.1 Application manager with Scheduler

Application Manager will provide Scheduler with the application and the meta file of the application, where information related to scheduling are present (start and end times, job type etc)

4.4.2 Application manager with Monitoring and Fault tolerance

- The monitoring and fault tolerance module interacts with Application manager, Authentication Manager and Analytic module.
- For all these modules, periodic monitoring will be provided through heart beat message.

4.4.3 Application manager with Sensor Manager

Sensor manager will provide sensor's information to the application manager which will be stored at application database along with other application information and passed to scheduler in form of meta data with application's other information.

4.4.4 Application manager with Controller Manager

Control manager will send Controller's information to the application manager and that information will be stored at application database along with the other app information.

4.5 Interactions between sub-modules

4.5.1 User-Interface with Authentication module

- UI will take username and password pass to authentication module to authenticate it.
- After authenticating it, authentication module will tell UI to show the message about successful or unsuccessful Authentication.

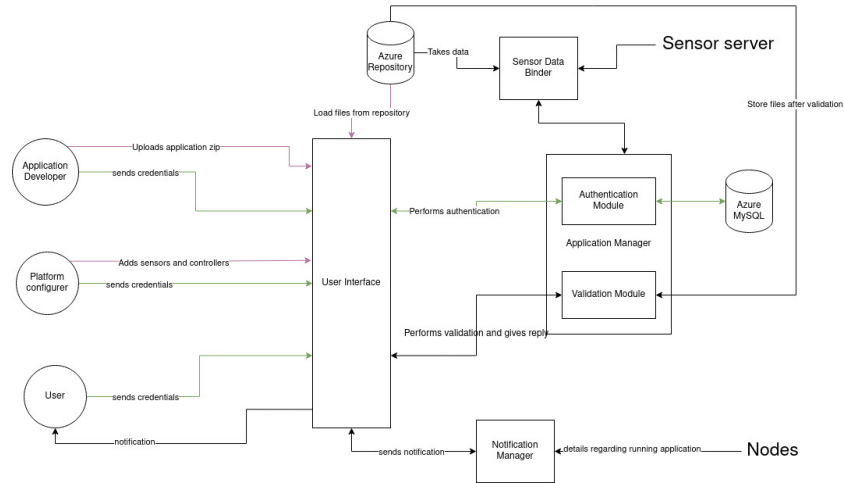


Figure 4: Block Diagram

4.5.2 User-Interface with Validator

- UI will take zip file and pass to validator to validate according to specific format.
- After validating it, validator module will tell UI to show the message about Valid or invalid zip file.

4.5.3 Application manager and Sensor data binder

- Application manager will pass information about application and sensor's data after authenticating and validating application zip file.
- Sensor data binder will bind sensors and application together which application will use sensors.

4.5.4 User-Interface and notification manager

- Notification manager will send notification to each user's UI if notification is generated by nodes.

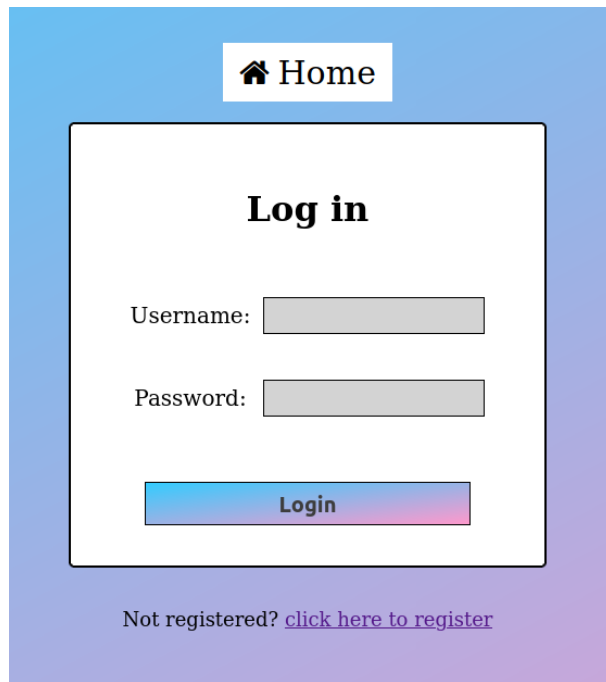
5 Output



Figure 5: Platform Homepage

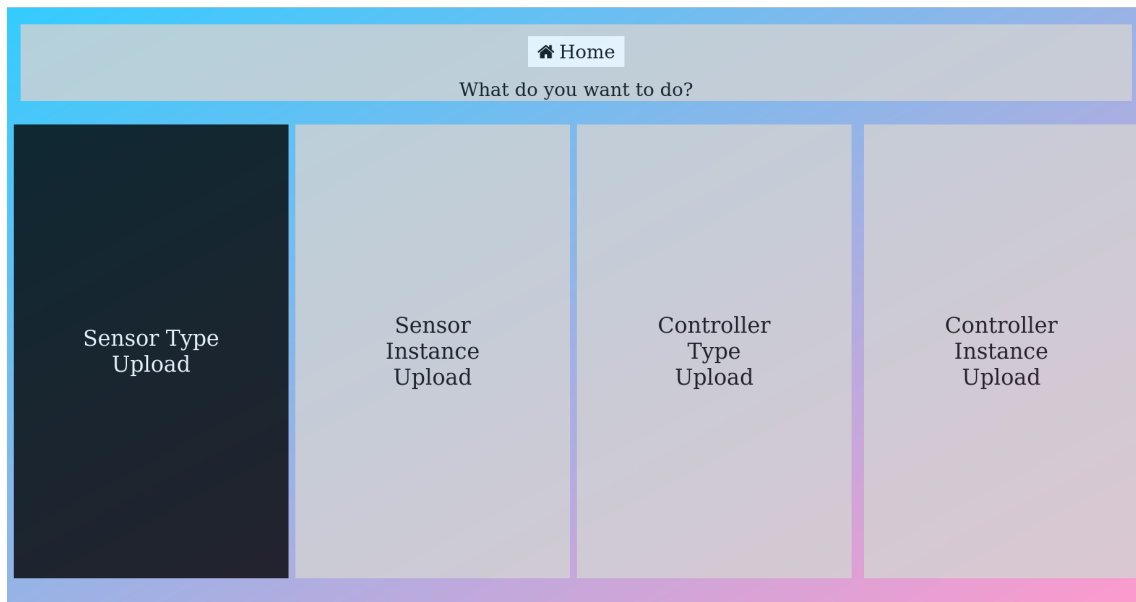
The screenshot shows the 'Registration' page. At the top, a 'Home' link with a house icon is visible. The main content area is a white box with a black border containing the title 'Registration'. Below the title are two input fields: 'Username:' and 'Password:'. A 'Register' button is positioned below these fields. At the bottom of the white box, there is a link that reads 'Already registered? [click here to login](#)'.

Figure 6: Registration page



The login page features a blue gradient background. At the top center is a white button with a house icon and the text "Home". Below this is a white rectangular box with a black border. Inside the box, the text "Log in" is centered in bold. Below the title are two input fields: "Username:" followed by a gray text box, and "Password:" followed by a gray text box. Below the password field is a blue button with a gradient and the text "Login". At the bottom of the white box, the text "Not registered? [click here to register](#)" is displayed, with the link in purple.

Figure 7: Login page



The platform configurer homepage has a blue gradient background. At the top is a light blue header bar containing a white button with a house icon and the text "Home". Below the header is a gray bar with the text "What do you want to do?". The main content area consists of four vertical panels. The first panel on the left is dark blue and contains the text "Sensor Type Upload". The second panel is light blue and contains the text "Sensor Instance Upload". The third panel is light blue and contains the text "Controller Type Upload". The fourth panel is light blue and contains the text "Controller Instance Upload".

Figure 8: Platform configurer homepage

Home

Appinfo(id, Name):

App id:

Location Information

City:

Do you want to deploy at specific Location? Yes: ☐ No : ☒

Execution Information

Date:

dd / mm / yyyy

Start Time:

-- : -- --

Duration(in minutes):

Repeataation needed? Yes: ☐ No : ☒

Submit

[Click here to go back to User home page](#)

Figure 9: Deployment page(shrunked)

Home

Appinfo(id, Name):

App id:

Location Information

City:

Do you want to deploy at specific Location? Yes: ☒ No : ☐

Room no:

House No:

Street:

Execution Information

Date:

dd / mm / yyyy

Start Time:

-- : -- --

Duration(in minutes):

Repeataation needed? Yes: ☒ No : ☐

After how much time?

Year

0

Month

0

Day

0

Hour

0

Minute

0

Submit

[Click here to go back to User home page](#)

Figure 10: Deployment page