**IIIT\_H/ Spring 2015**

**Internals of Application Systems**

**Project Module 3 : Registry, Repository and Security**

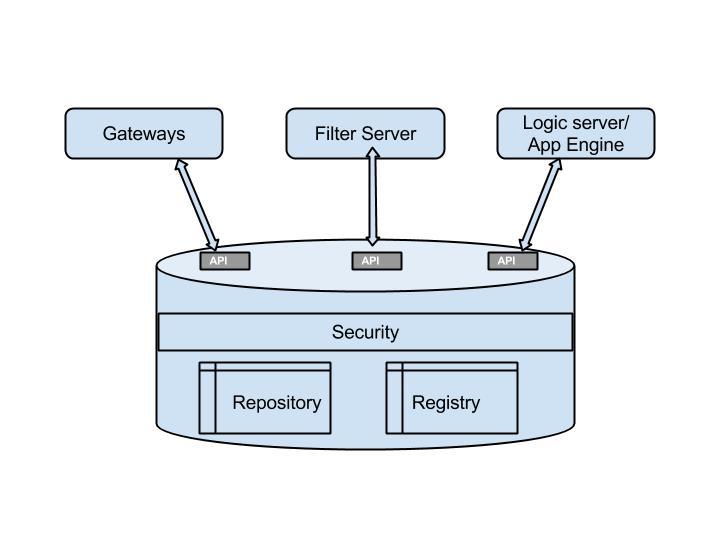
1. **Project Introduction**

This project aims to build a sensor network based on IoT platform and develop an interactive android application that receives multiple type of data from sensor network. This application shall allow users to view information such as temperature, traffic or sensor type data, generate reports and control related devices from their android device.

Following are overall modules of this project

1. Sensors, Gateways, and communication.
2. Filter server , App engine and client app
3. **Registries, Repository and Security**
   1. **Functional overview of Team’s module**

This module fulfills the data storage requirements and aids security in overall IoT platform. Repository is the static data unit which stores the identity, protocol and handler information of various devices ( sensors, gateways, filter server). Registry is the dynamic data unit which stores the data received from gateways. Security is achieved by ensuring that only registered devices can communicate with platform.



1. **Test cases**
   1. Test cases- used to test the team’s module
2. Database boot up: Set up all the required tables using script and schema files.

Input: Start up script, configuration file, schema files

Expected behavior: Required database, tables should be created with correct schema

1. Repository initialization: Static information loading in repository at boot time from a json/xml file containing the list of devices and connected sensors

Input: Json/xml file

Expected behavior: Repository table should be set up with correct static configuration information as given in input file

1. Data Integrity test: System shall not allow ids of two or more sensors, gateways to be same.

Input: try to add two sensors with same id in configuration file

Expected behavior: Duplicate entry is discarded and error is reported.

1. Consistency test: Devices that are listed under repository are the only ones that can appear in corresponding columns in registry. For example if a new sensor is up, but it is not yet listed in repository, sensor data shall not be updated in registry.
2. Missing required data test: If any incoming call for new data entry has missing required field value, error should be reported to the sender and entry shall be discarded.
3. *Performance test*

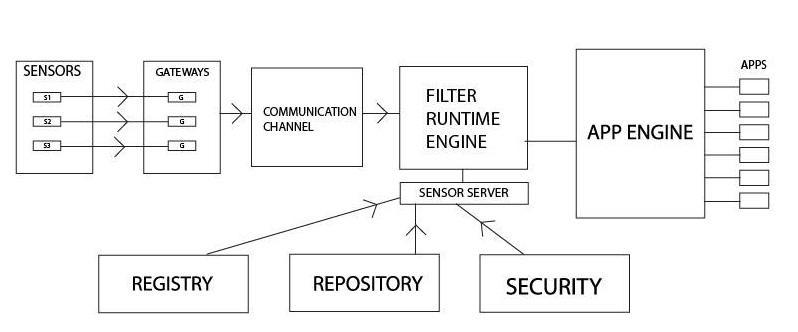
Input : Normal Load : average requests to the database server

Heavy Load : large number of requests to the server

Expected Output : Server should handle load correctly and should not get overwhelmed by heavy load.

1. Security test

* Data from unauthorized sensor: Add a new sensor in the system which is not listed under repository. System shall not allow data from any unauthorized sensor to be added.
* Data from unauthorized gateway: Try to send data from a gateway which is not listed under repository. System shall not allow data from any unauthorized gateway to be added.
  1. Overall project test cases (relevant to the module)

1. Accessibility from gateway: Gateways shall be able to connect and update registry using the API exposed for gateway to database communication.
2. Accessibility from filter server: Filter server shall be able to connect to and fetch data from registry using the API exposed for filter server to database communication.
3. Accessibility from logic server: Logic server shall be able to connect to and fetch data from registry using the API exposed for logic server to database communication
4. **Solution design considerations** 
   1. Design big picture
   2. Environment to be used

* MongoDB , Apache server
* To store registry and repository tables, Interface to access/update tables, authentication
* RPC / HTTP(S) for communication
  1. Technologies to be used
* JSON, XML
* Configuration files, static data
  1. Devices registry & repository

*Registries:*Registry is the dynamic data unit which stores the data received from gateways at runtime.

*Repository:*Repository is the static data unit which stores the identity, protocol and handler information of various devices ( sensors, gateways, filter server).

* 1. Communication overview

Communication will be done using Sockets , RMI or REST API depending upon appropriate resource availability.

* 1. Interactions between modules
  + Gateway to registry & repository: When a gateway boots up, it retrieves its configuration information from repository. This configuration information includes list of reachable sensors, gateway type handlers and protocol information.
  + Filter server to registry & repository: Filter server queries registry to get the records from specific type of sensors.
  + Logic Server/ App Engine to repository: Before registering itself with filter server, logic server fetches the list of sensors from repository based on GPS co-ordinates and range.
  1. Wire and file formats

Repository loader format: XML

<gateway id>

<sensor id type>

<device handler>

<protocol>

1. **Key data structures**

Solution level data structures

* Repository

gateway id, sensor id, sensor type, device handler, protocol

* Registry

gateway id, sensor id, last downtime, last update time, location, value, checksum

Persisted data (registry, rules, et al)

Registry database

Rules Database

1. **Interactions & Interfaces**

APIs

isValidSensor (sensor id) : Filter server or Logic server can check if the sensor is a authorized one or not.

isValidGateway (gateway id) :Filter server or Logic server can check if the gateway is a authorized one or not.

addNewSensor( sensor id, sensor type, device handler, protocol) : To add a new sensor device in the repository.

addNewGateway ( gateway id, gateway address) : To add a new gateway device in the repository.

getSensorList (GPS location, range) : Called by logic server to get the list of sensors in given area.

getSensorData ( sensor type, from time, to time): Called by filter server to fetch specific type data from registry.