Internals of Application Servers

User Guide

Submitted By:
Group 2

Machine Requirements:

- **MongoDB:** It is an open-source document database, and leading NoSQL database. It avoids traditional table-based relational database structure in favour of JSON-like documents with dynamic schemas . Since our application platform uses MongoDB as the backend database, we need MongoDB to be installed on the development system.

- **Node.js:** It is a platform built on **Chrome's JavaScript runtime** for easily building fast, scalable server-side and networking applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices. Node.js will be to produce dynamic data through the sensors. It is used for communication throughout our platform.

- **Java:** Since the application are basically built on Android Platform, we require Java to be present in the system.

- **Android SDK:** It is software development kit that enables developers to create applications for the Android platform. We require the SDK to be present in the system so as to be able to develop applications that benefit from the platform.

- **Raspberry Pi System:** Raspberry Pi will be used as the gateway that communicates/gets data from sensors, stores them in the database and then passes on this information to the filter server for further processing.

DATA FORMAT:

- **Data Storage in MongoDB:** MongoDB accepts and retrieves records as JSON/BSON objects. A given mongo database is broken up into a series of BSON files on disk with increasing size up to 2GB. BSON is its own format, built specifically for MongoDB.

- **Data Retrieval from Database:** Read operations, or queries, retrieve data stored in the database. The data retrieved from the database will be in JSON format so developer of the application needs to be careful while dealing with the data format. He should be able to decompress and present data in such a way that is understandable by the user application.

- **Data parsing in the app being built:** The gateway will forwards the data to the filter server via TCP/IP protocols. The filter server will check for the validity of the data and any duplicate or invalid data is neglected. The Sensor registry table will also be accessed and information will be mapped to the corresponding sensor id. Now the developer can identify the fields in which he is interested and can use the data corresponding to that field for further operations. For example if he is interested in analysing temperature he can only use the data forwarded by the temperature sensors.

- **Parsing Json data returned from the API:** The first step is to identify the fields in the JSON data in which you are interested in. For example. Select temperature if we interested in getting temperature only. For parsing a JSON object, we will create an object of class JSONObject and specify a string containing JSON data to it. An JSON file consist of different object with different key/value pair etc. So JSONObject has a separate function for parsing each of the component of JSON file. Some methods provided by JSONObject class for better parsing JSON files are:

  - **get(String name)**
  - **getBoolean(String name)**
  - **getDouble(String name)**
  - **getInt(String name)**
  - **getLong(String name)**
  - **length()**
  - **names()**

- **Call API to register sensor data and store it:** All the sensors will be allotted a unique id. A table will be maintained in database in which each sensor will be associated with unique identification id, upper/lower threshold and the location of the sensor so that when the server receives information from any of the server it searches the table for location corresponding to the sensor id.

Configuration:

1. **Configuring the Gateway** - Connect the sensors to the gateway and verify the sensor data properly reaches the gateway. Also verify whether the gateway is properly communicating the data to the backend database as well as the filter server.
2. **Configuring the Filter Server** – Filter Server also acts as the rules engine to determine proper values from the sensor that needs to be sent to user application. Configure and connect this

to the gateway as well as the database that stores the sensor values received from the gateway.

3. **Sensor Devices** – Configure the sensor devices to properly communicate with the gateway and understand the type of data they send.

Developer Module:

The developer working on this platform will be able to use the APIs provided by the platform to build applications that cater to the needs of the users. The application can be used to view sensor data of a particular location in real time. The APIs provide all the controls to add, delete, modify sensor information as well as retrieve data from the sensors.

1. **Sensor related APIs**: These APIs help the developer to add, delete or modify sensor information. Since the application platform is dynamic in nature, the sensors can be added in real time.
2. **Data Retrieval APIs**: These APIs will help the developer retrieve the data of the corresponding sensor. The sensor data then can be used to display real-time information to the user.