**Application Overview:**

1. Signals from various IOT devices will be recorded in a single file in .JSON format.
2. For each IOT device, rules will be defined by Users.
3. If any IOT device behaves differently and matches with the rules provided by user, then this application will trigger message by indicating the IOT signal Name, Signal Value and User Rule.
4. Using this program output, user will be able to monitor the IOT devices easily.

**Files & Description:**

1. FN\_ParseJSON.sql -> This file is used to convert JSON file into SQL Table format. This function can be replaced by ParseJSON, which is an inbuilt SQL Server function.
2. SP\_RaiseSignal.sql -> This file is used to trigger warning message to user if any of the IOT device violates the rules.
3. SP\_RaiseSignal\_UT.sql -> This file is used to test the SP\_RaiseSignal Stored Procedure.
4. SQLExecution.sql -> This file is used to execute the SP\_RaiseSignal.sql file
5. SQLExecution\_UT.sql -> This file is used to execute the SP\_RaiseSignal\_UT.sql file
6. ExecuteSQL.bat -> (Change the file format from .txt to .bat) This file is used to invoke the SQLExecution.sql from Command window and returns the result in SignalOutput.txt file.
7. ExecuteSQL\_UT.bat -> (Change the file format from .txt to .bat) This file is used to invoke the SQLExecution\_UT.sql from Command window and returns the result in SignalOutput\_UTResult.txt file.
8. Raw\_signal.json -> this is the collected data from various IOT devices and input json file to our application.
9. Raw\_Signal\_UT.Json -> this file is used for Unit testing purpose. To unit test this application, the Input file should be like this.
10. Signal\_rules.json -> This file is created based on user rules for each IOT device.

**Architecture:**

1. User rules will be captured using User Interface or in JSON file.
2. Those rules will be saved in Database, in the above mentioned (Signal\_rules.json) format.
3. Signals from IOT devices might contain 1000s of data. So validating those signals in the middle ware of the application is not an efficient way. It leads to performance problem.
4. So created the entire validation logic in the Database Procedure.
5. Since the entire logic is in Database, output can be stored in the DB directly.
6. Just for additional option, I exported the data in .txt format.

**Limitations:**

1. I used Table variable in the Stored Procedures. So it can support maximum of 1000 records. If requires, it can be replaced with static tables to overcome this issue.
2. Since I used SQL BLOB data type to store input JSON data, this code will not support, if the input file size goes beyond 2 GB. Again this can be eliminated, if we use middle ware application created in .net/python/java to collect data and store it in Database.
3. Performance could be improved if we use SQL Server 2016 version or parsing the JSON file in the Middleware. But need not to change the architecture.

**Assumptions & Difficulties:**

1. User input will be collected in the above mentioned Signal\_rules.JSON file format.
2. The solution which I provided is not an end-to-end solution but solves the key problem of this application.
3. Since my personal laptop is at my hometown and cannot execute the complete code in the system, which I’m using, I’m unable to test the code.
4. The system which I’m using has SQL Server 2012 version and no Visual Studio. So I couldn’t create some User Interface to accept user input or parse JSON data directly. And created a custom function to Parse JSON data.
5. If it is fine to provide code by weekend, I can test the code in my personal laptop and send it again.

**Execution Steps:**

1. Execute the FN\_ParseJSON.sql file in Microsoft SQL Server.
2. Copy the Raw\_signal.json, Raw\_signal\_UT.json and Signal\_rules.json in the drive.
3. Open the file SP\_RaiseSignal.sql and replace the drive path in the <FileLocation>.
4. Open the file SP\_RaiseSignal\_UT.sql and replace the drive path in the <FileLocation>.
5. Execute the SP\_RaiseSignal.sql and SP\_RaiseSignal\_UT.sql

If you want to execute the query directly in SQL Server Management Studio, then no need to follow the below steps.

1. Execute the SQLExecution.sql and SQLExecution\_UT.sql files.
2. Execute ExecuteSQL\_UT.bat file for Unit Testing
3. Execute ExecuteSQL.bat file for Output.