Software Developer Cloud Ops Intern

An Internship Report

Submitted in partial fulfillment of the requirements for the award of a degree of

Bachelor of Technology

In

Computer Science

Submitted to

From 03/10/2023 to Present

SUBMITTED BY

Name – Rakesh Roshan
Reg. No. – 12011028
Company – OpenText
Project – EPI Rule Management
Submitted to – Neha
HOD – Dr. Virat (virrat.14591@lpu.co.in)

Student Declaration

To whom so ever it may concern

I, Rakesh Roshan, 12011028, hereby declare that the work done by me as a Software Developer Cloud Ops Intern from October 3rd, 2023 to Present, under the supervision of Technical Lead Mr. Naveen Kumar Kandasamy and Manager Mr. Ankur Jain at OpenText is a record of original work for the partial fulfillment of the requirements for the award of the degree, Computer Science and Engineering.

Rakesh Roshan
(12011028)

Dated: 01-03-2024

EPI Rule Management Project

Problem Statement:

The project aims to create a user-friendly web application for OpenText's EPI Rule Management, allowing customer support owners to efficiently manage XML rule files through a graphical user interface (GUI). The application should expose XML rules and their outputs, provide validation, testing, and GitLab integration, and include comprehensive user documentation.

Project Tasks:

1. GUI Development: Design and develop a web-based GUI for managing XML rule files.

2. Exposing XML Rules: Display all XML rules in the frontend UI and their corresponding output.

3. Validation Feature: Implement a validation button to check XML syntax.

4. Testing Feature: Include a test button to validate rules using user-input arguments and display matched rules.

5. GitLab Integration: Incorporate a Git_Commit button to push validated and tested rule files to GitLab.

6. User Documentation: Create comprehensive user documentation and help resources for effective application usage.

Success Criteria:

1. User-Friendly GUI: The application must have a user-friendly and intuitive interface.

2. Functional Buttons: Each button (Validation, Test, Git_Commit) must work as intended.

3. Comprehensive Documentation: The user documentation should be detailed and guide users effectively.

Understanding About the Project:

1. GUI Development and User Interaction:
   - The GUI should provide an intuitive interface for users to view, validate, and update XML rule files. It should include features like dropdown menus, input fields, and buttons for easy navigation and interaction.

2. Validation and Testing Features:
   - The application should include buttons or options for users to validate the XML syntax and test the rules with sample input values. It should provide feedback on the validity and correctness of the rules.

3. Git Integration:
   - The application should integrate with GitLab to allow users to commit, and push validated and tested rule files to a specified repository. This feature should include options for users to input GitLab credentials and repository details.

4. User Documentation and Help Resources:
   - The application should include comprehensive user documentation and help resources to guide users on how to effectively use the web application. This documentation should cover topics such as navigating the GUI, validating rules, testing rules, and pushing files to GitLab.

5. Success Criteria:
   - The application must meet the following success criteria:
     - The GUI must be user-friendly and easy to use.
     - Each button must be functional and must meet its intended purpose.
     - The user documentation and help resources must be comprehensive and must guide users in effectively using the web application.

Requirements:

1.Languages and Frameworks used: HTML, CSS, JavaScript, XML, Groovy, Node.js

2. Knowledge of Git Version Control System and GitLab

WORKING OF THE PROJECT:

Here’s a more detailed set of instructions:

1. Clone the Repository:

Clone this repository into your local system:
git clone https://gitlab.otxlab.net/obm/xml_generator.git

2. Install Node.js and npm:

Install Node.js and npm on your Linux terminal:

```
sudo apt update
sudo apt install nodejs
sudo apt install npm
```

3. Install Java:

Check if Java is installed:

```
java -version
```

If not installed, install Java using the package manager:

```
sudo apt install default-jre sudo apt install default-jdk
```

4. Install Groovy:

Install Groovy using the package manager:

```
sudo apt install groovy
```

5. Set the JAVA_HOME Environment Variable for Groovy:

Find the Java installation directory:

```
sudo update-alternatives --config java
```

Open the shell configuration file in a text editor (such as .bashrc or .zshrc):

```
nano ~/.bashrc
```

Add the following line at the end of the file, replacing /path/to/your/java with the actual path you noted in step 5:

```
export JAVA_HOME=/path/to/your/java
```

Save the file and exit, then apply the changes:

```
source ~/.bashrc
```

Verify the Groovy installation:

```
groovy --version
```

6. Navigate to the Project Directory:

Change to the project directory:

```
cd xml_generator
```

7. Install Project Dependencies:

Install project dependencies using npm:

```
npm install
```

8. Start and Run the Node.js Server:

Start and run the Node.js server:

```
node server.js
```

9. Access the Node.js Web Application:

Open a web browser and navigate to http://localhost:3001 (or the specified port in server.js).

SOLVED FAQ’s

FAQ for EPI Rule Management Project:

1. **Issue**: Validation of XML rule files resulted in inaccuracies.

- **Solution**: The inaccuracies may have arisen due to manual data entry, which can sometimes lead to human error. By automating this process within the GUI, we minimize such errors and ensure the accuracy of the data.

2. **Issue**: During the XML rule validation, the data was only being inserted into the middle column.

- **Solution**: The functionality has been enhanced by employing basic mathematical operations within the GUI. This allows us to modify the target location for data extraction, enabling us to insert data into any desired row or column. This enhancement provides greater flexibility and control over the data manipulation process.

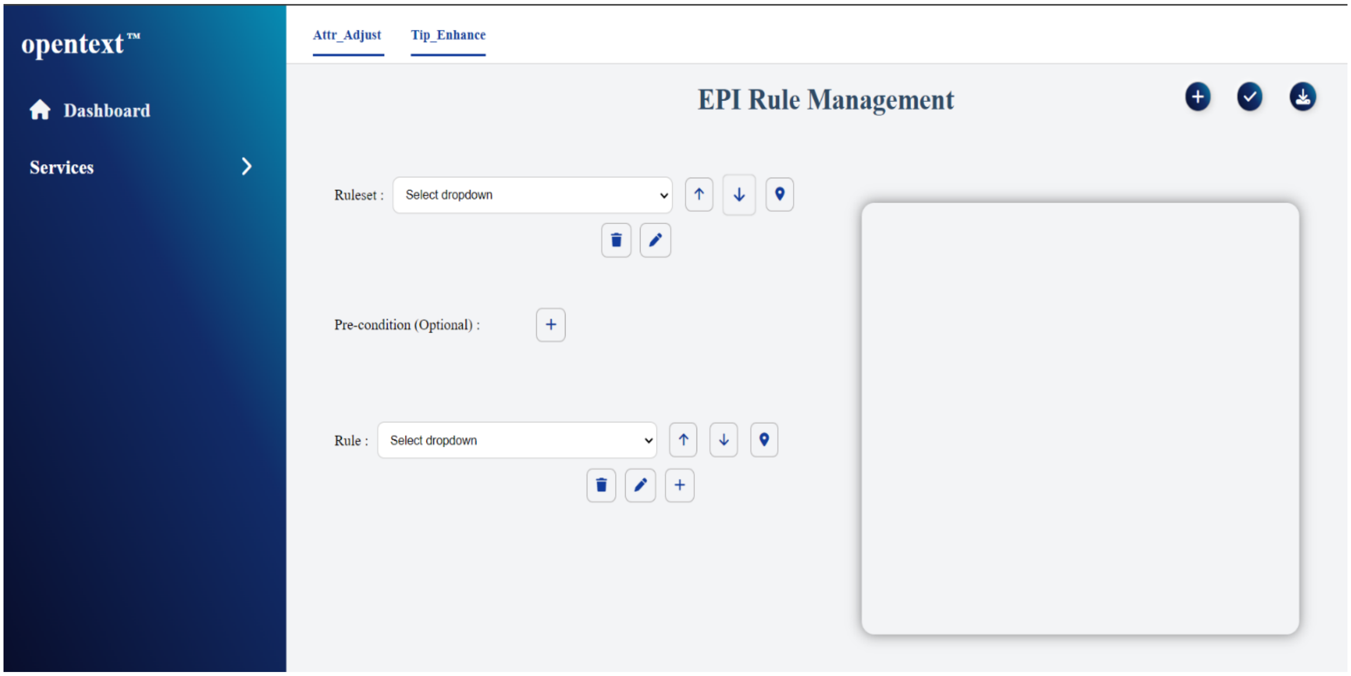
3. **Issue**: The program was not terminated as expected with a keyboard interrupt (Ctrl+C) during the validation process.

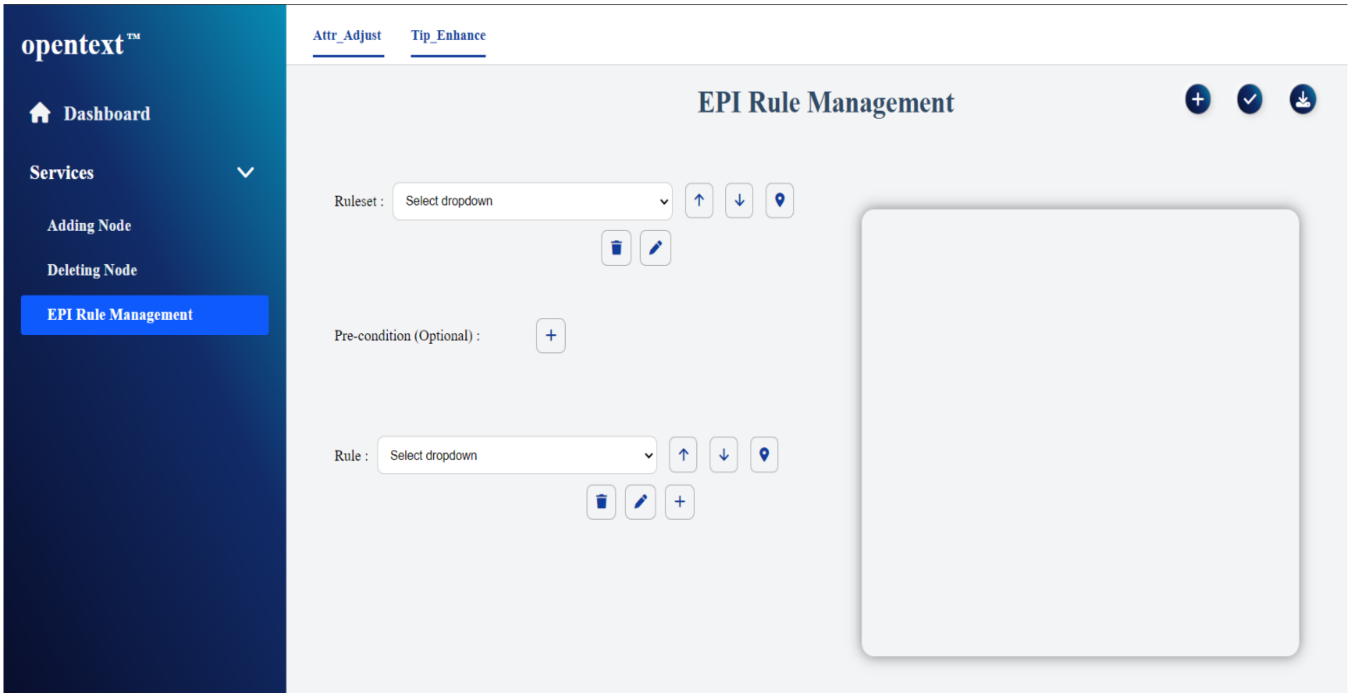
- **Solution**: We enhanced the code by incorporating an exception handler for keyboard interrupts. This ensures the program terminates gracefully upon a Ctrl+C command within the GUI.

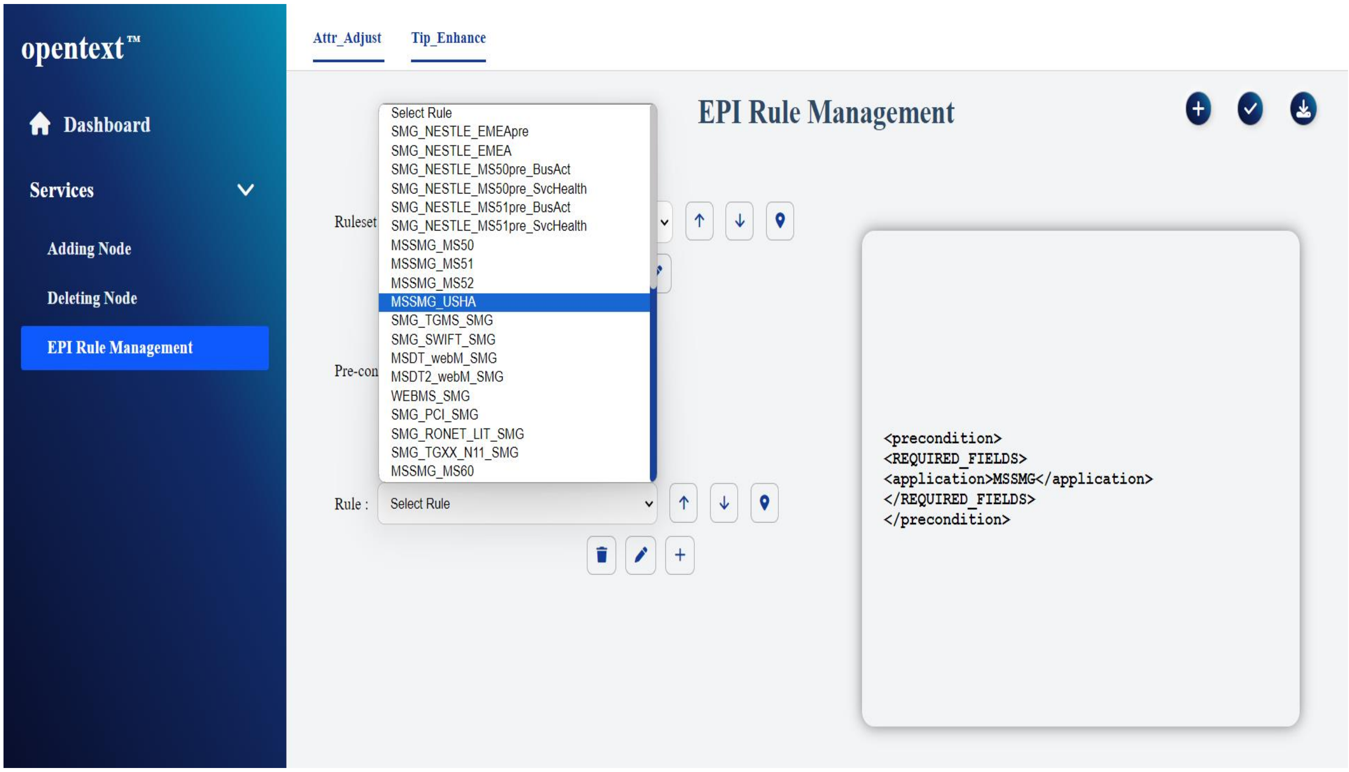
FUTURE SCOPE

To tailor the project to the specific needs of the employees it is proposed to enhance the project by incorporating a Graphical User Interface (GUI). The primary objective of this modification is to safeguard the integrity of the code, preventing unauthorized alterations. Additionally, the introduction of a GUI will significantly improve the user interface, making it more intuitive and user-friendly. This approach ensures that the project remains secure, while also enhancing its usability.

SCREENSHOTS

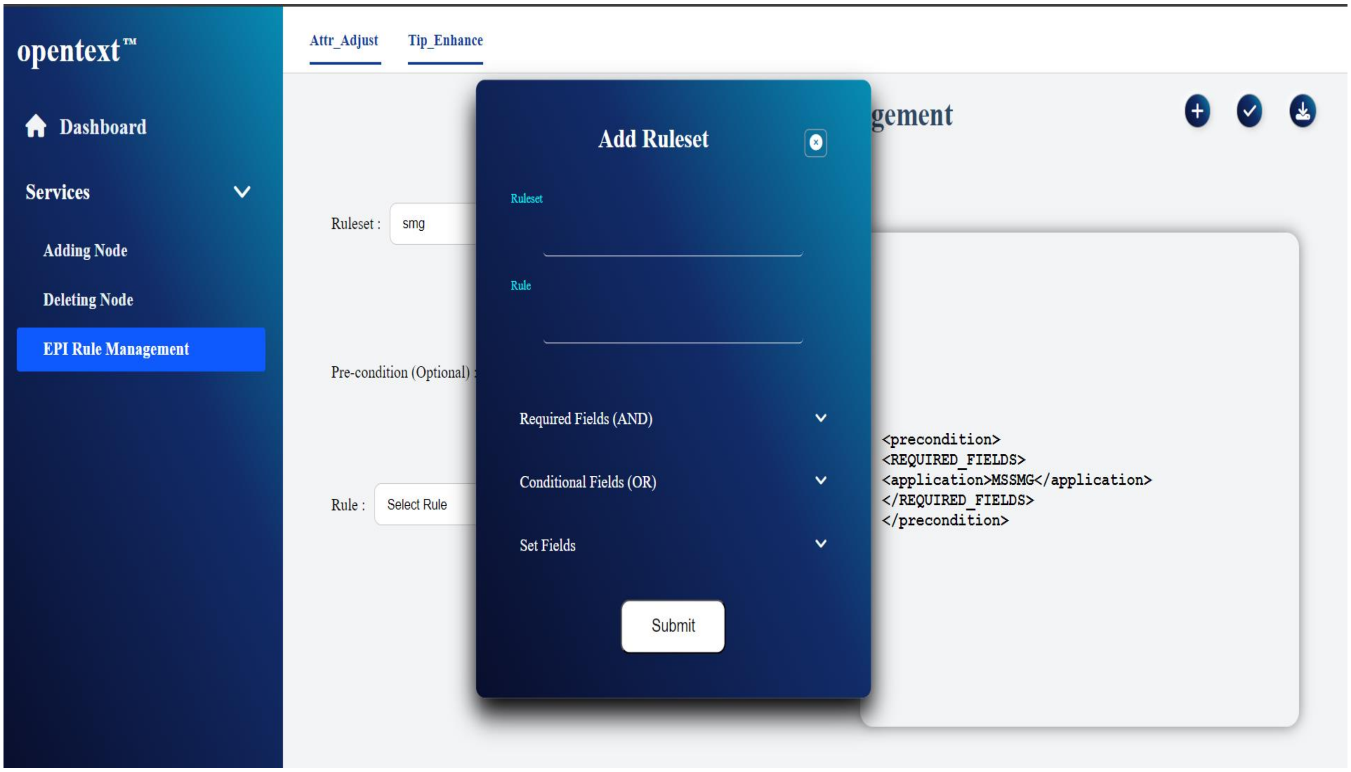






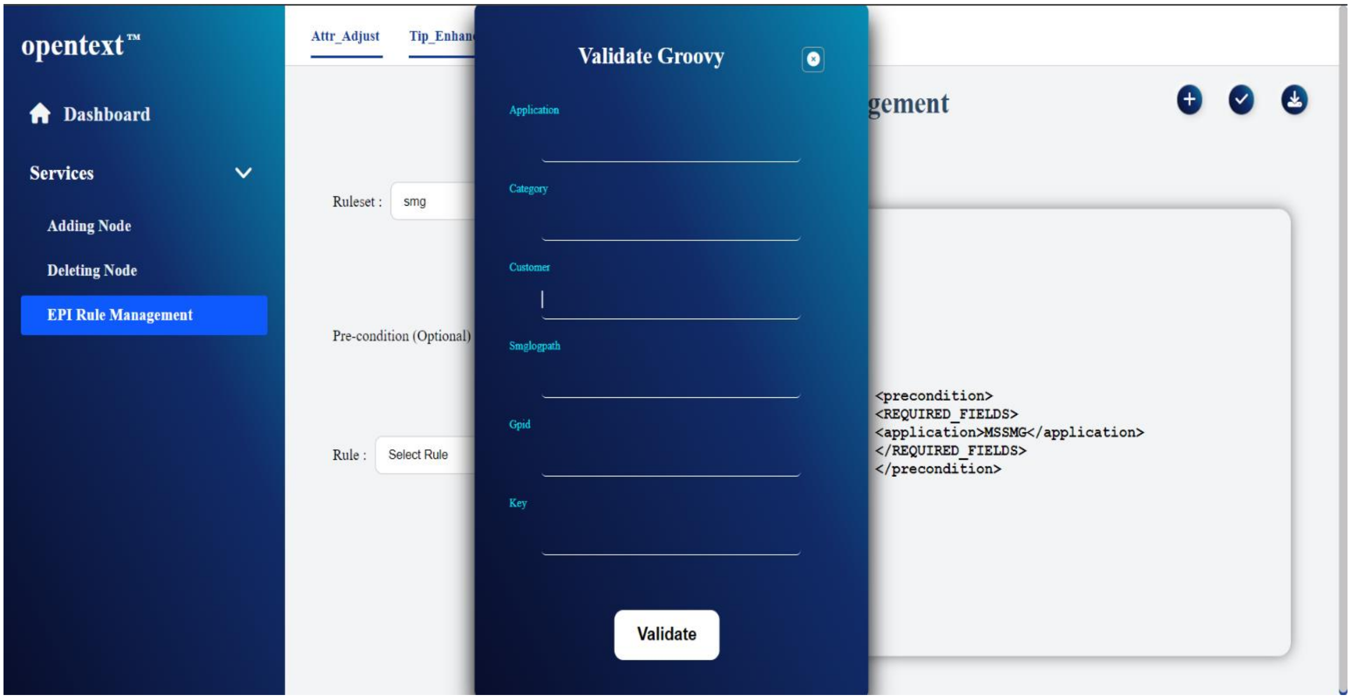
opentext™
Dashboard
Services
Adding Node
Deleting Node
EPI Rule Management
Attr_Adjust
Tip_Enhance
EPI Rule Management
Ruleset
Pre-con
Rule :
Select Rule
Select Rule
SMG_NESTLE_EMEApre
SMG_NESTLE_EMEA
SMG_NESTLE_MS50pre_BusAct
SMG_NESTLE_MS50pre_SvcHealth
SMG_NESTLE_MS51pre_BusAct
SMG_NESTLE_MS51pre_SvcHealth
MSSMG_MS50
MSSMG_MS51
MSSMG_MS52
MSSMG_USHA
SMG_TGMS_SMG
SMG_SWIFT_SMG
MSDT_webM_SMG
MSDT2_webM_SMG
WEBMS_SMG
SMG_PCI_SMG
SMG_RONET_LIT_SMG
SMG_TGXX_N11_SMG
MSSMG_MS60

```
<precondition>
<REQUIRED_FIELDS>
<application>MSSMG</application>
</REQUIRED_FIELDS>
</precondition>
```



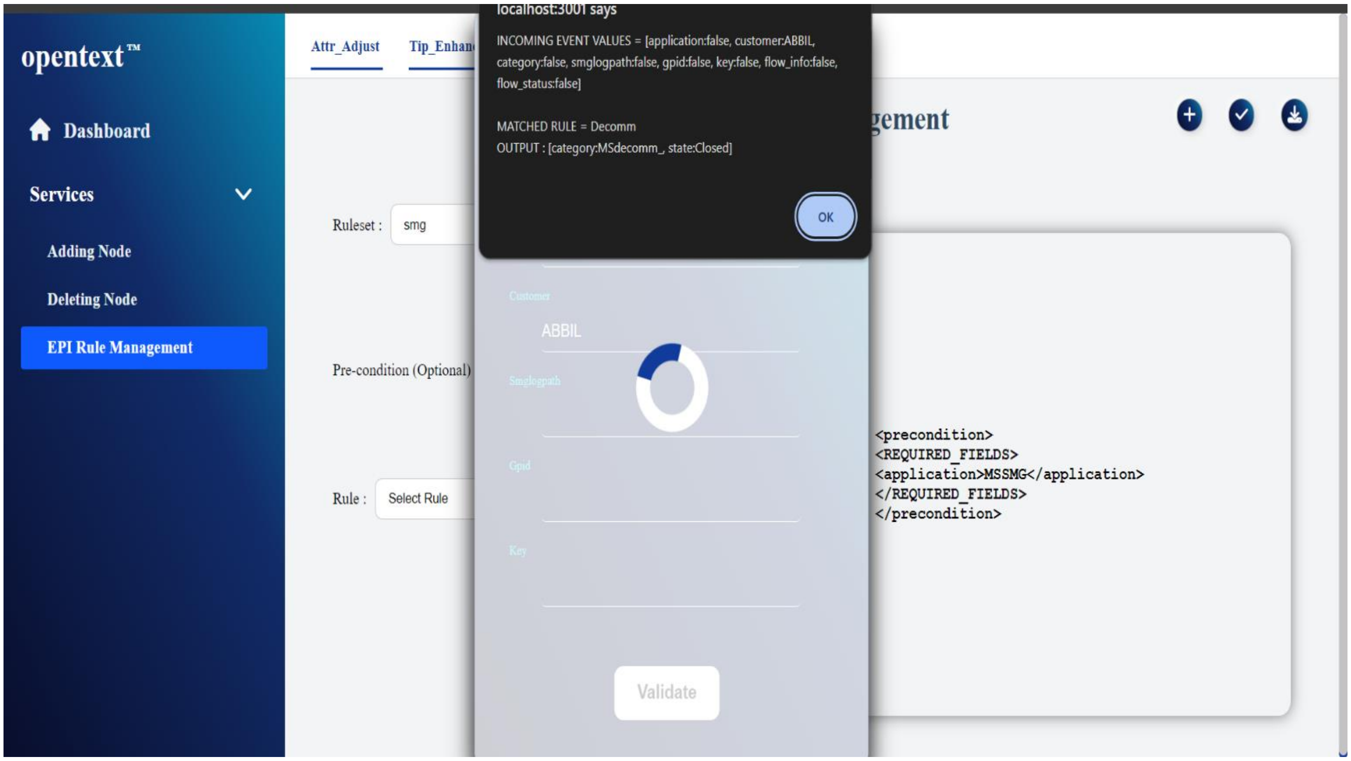
opentext™
Dashboard
Services
Adding Node
Deleting Node
EPI Rule Management
Attr_Adjust
Tip_Enhance
gement
Ruleset :
smg
Pre-condition (Optional) :
Rule :
Select Rule
Add Ruleset
Ruleset
Rule
Required Fields (AND)
Conditional Fields (OR)
Set Fields
Submit

```
<precondition>
<REQUIRED_FIELDS>
<application>MSSMG</application>
</REQUIRED_FIELDS>
</precondition>
```



Ruleset : smg

```
<precondition>
<REQUIRED_FIELDS>
<application>MSSMG</application>
</REQUIRED_FIELDS>
</precondition>
```



Ruleset : smg

Customer ABBIL

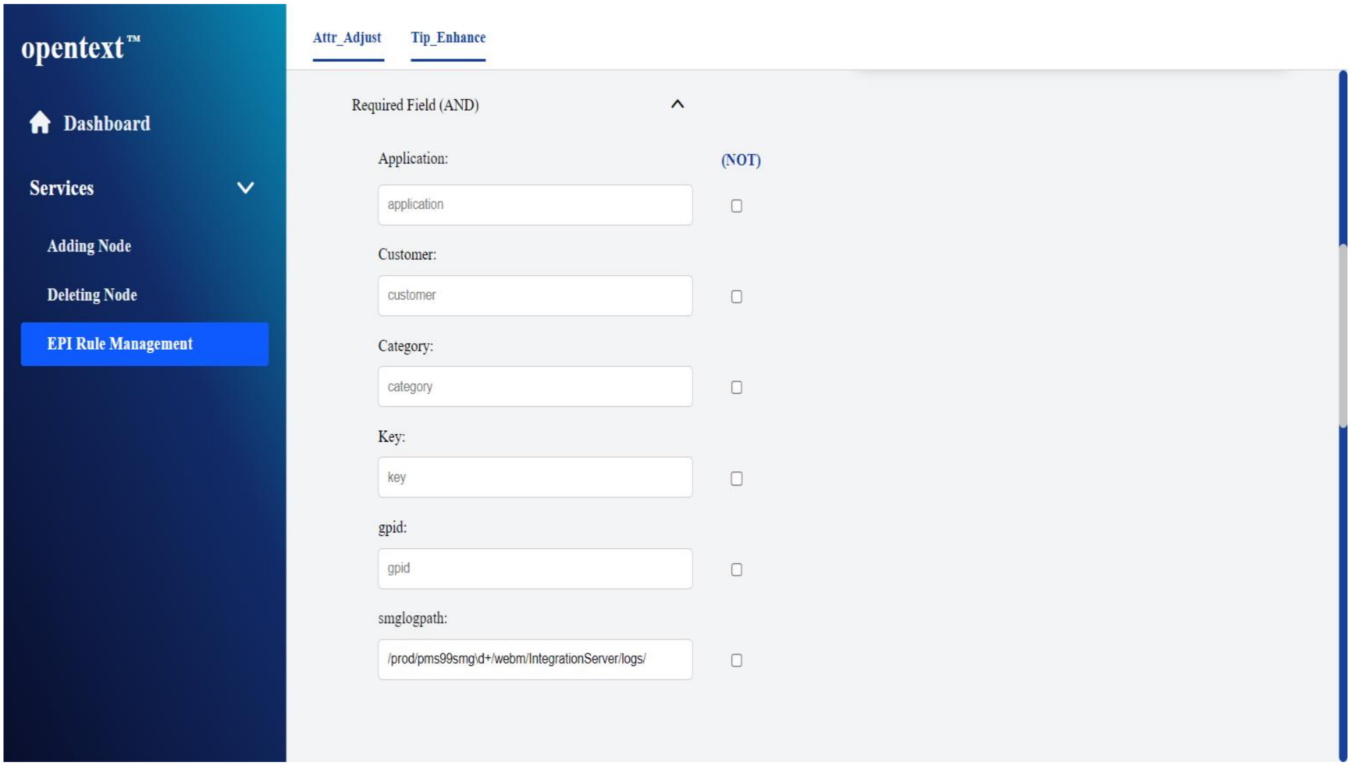
```
<precondition>
<REQUIRED_FIELDS>
<application>MSSMG</application>
</REQUIRED_FIELDS>
</precondition>
```

localhost:3001 says

INCOMING EVENT VALUES = [application:false, customer:ABBIL, category:false, smglogpath:false, gpid:false, key:false, flow_info:false, flow_status:false]

MATCHED RULE = Decomm

OUTPUT : [category:MSdecomm_, state:Closed]



Required Field (AND)

(NOT)

Application: application □

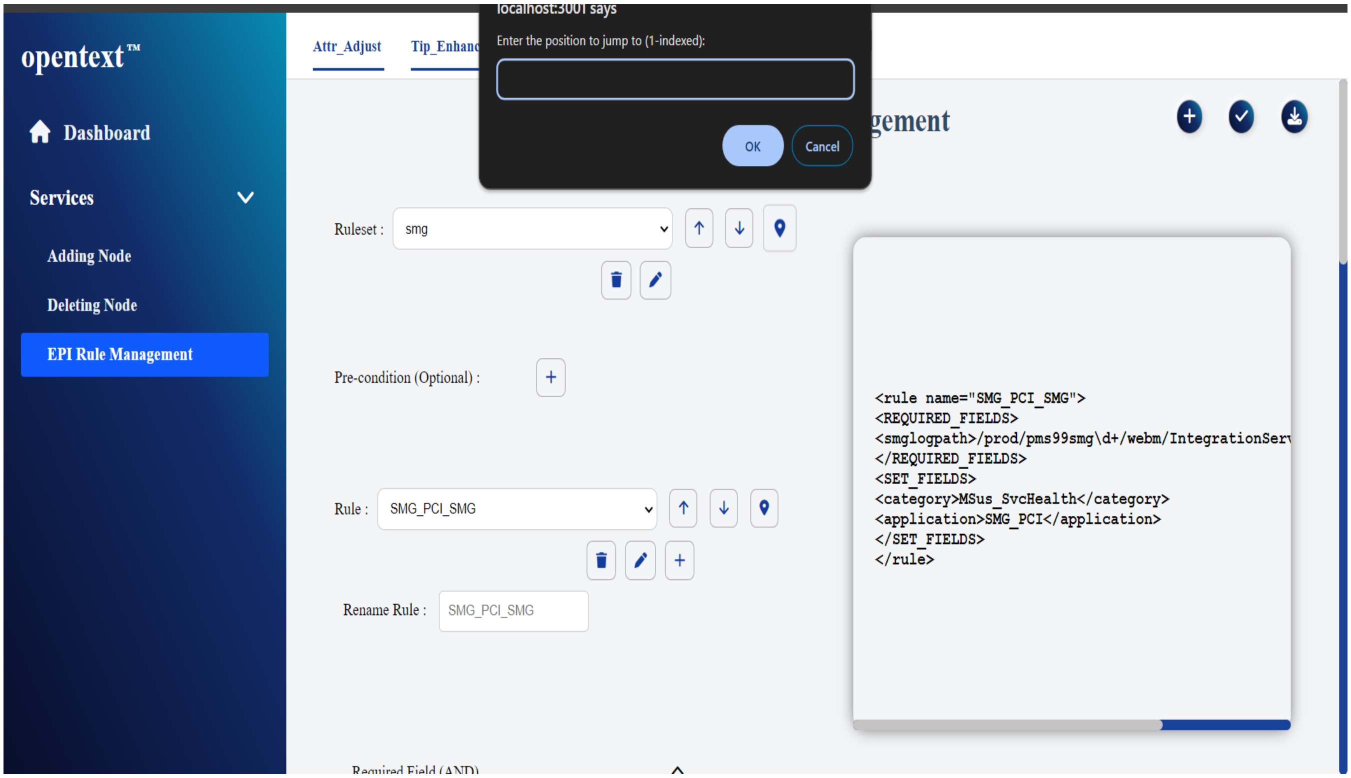
Customer: customer □

Category: category □

Key: key □

gpid: gpid □

smglogpath: /prod/pms99smg\d+/webm/IntegrationServer/logs/ □



localhost:3001 says

Enter the position to jump to (1-indexed):

Ruleset : smg

Pre-condition (Optional) :

Rule : SMG_PCI_SMG

Rename Rule : SMG_PCI_SMG

```
<rule name="SMG_PCI_SMG">
<REQUIRED_FIELDS>
<smglogpath>/prod/pms99smg\d+/webm/IntegrationServ
</REQUIRED_FIELDS>
<SET_FIELDS>
<category>MSus_SvcHealth</category>
<application>SMG_PCI</application>
</SET_FIELDS>
</rule>
```