Real-time Data Classification API

Introduction

Welcome to the Real-time Data Classification System! This project aims to create a powerful and dynamic platform for processing and classifying live data streams based on user-defined rules. It leverages modern web technologies and efficient backend solutions to ensure that data is processed with high speed and accuracy, making it ideal for applications that require real-time decision-making and data management.

Purpose and Goals :

The primary goal of this project is to provide a scalable, real-time data processing system where users can define their own classification rules. By integrating JWT for secure authentication, a robust database for rule storage, and real-time data streaming capabilities, this project ensures that user-defined rules can be applied promptly to incoming data, making the system suitable for various real-world applications.

Features

- JWT Authentication
  - Secure access with JSON Web Tokens.
  - Middleware to validate incoming requests.
  - Sign-up and login endpoints for user management.
- Database Integration
  - Schema design for storing user and rule data.
  - CRUD operations for managing user-defined classification rules.
  - Integration with a database for persistent storage.
- User-Defined Classification Rules
  - Parser to interpret and validate user-defined rules.
  - API endpoints for rule management (create, read, update, delete).
- Real-time Data Processing
  - WebSocket or Server-Sent Events (SSE) for live data streaming.
  - Real-time classification engine to process data streams using defined rules.
- Comprehensive Testing
  - Unit tests for individual modules.
  - Integration tests for end-to-end system validation.
  - Load testing to ensure system stability under high data volumes.

Tools & Technologies

- **Backend** : Fastify,NodeJS
- **Database** : Mongodb
- **Authentication** : Json Web Token (JWT)
- **Real-Time Processing** : WebSocket
- **Unit/Integration Testing** : Jest with Supertest

- **Load Testing** : Apache JMeter
- **API request Platform** : VS Code-Thunder Client

Installation

- **Clone the repository**
  git clone https://github.com/biswojit65/LDP-Project-3.git
  cd LDP-Project-3
- **Install dependencies using npm**
  npm install
- **Set up the database**
  Setup mongodb database and connect it to the application using mongoose.

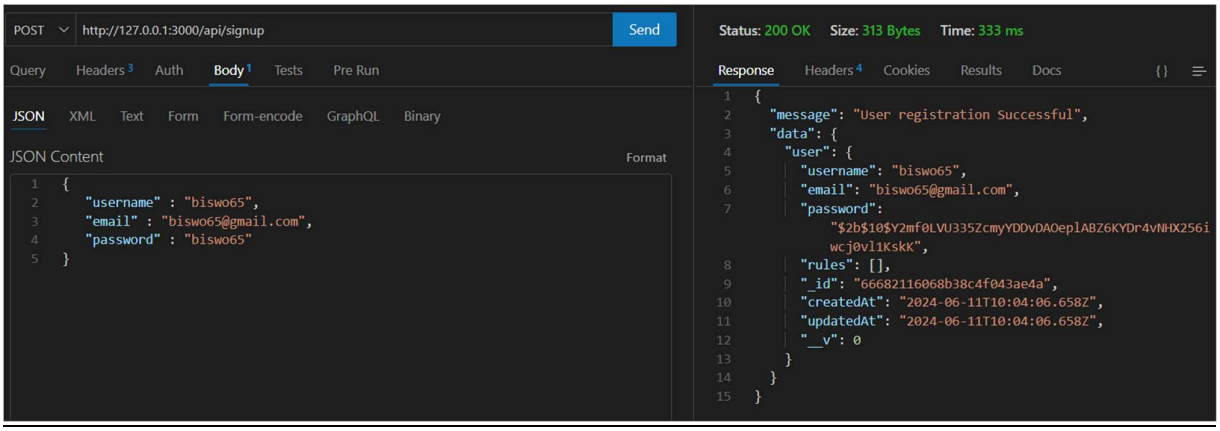
Usage

- **Start the server**
  npm start
- **API Documentation**
  You can use tools like Postman/VS Code Thunder Client to interact with the API endpoints listed below.

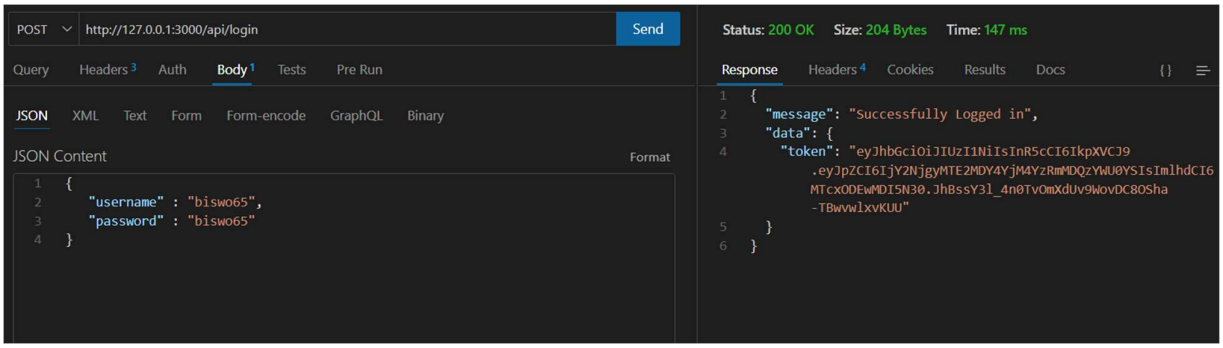
Endpoints

- **Create an User :** http://{{host}}:{{port}}/api/signup
- **Login a user :** http://{{host}}:{{port}}/api/login
- **Create a Rule :** http://{{host}}:{{port}}/api/newrules
- **Get All Rules :** http://{{host}}:{{port}}/api/getrules
- **Update a rule :** http://{{host}}:{{port}}/api/updaterule
- **Delete a Rule :** http://{{host}}:{{port}}/api/deleterule
- **Applying rules on inputstring :** http://{{host}}:{{port}}/api/check

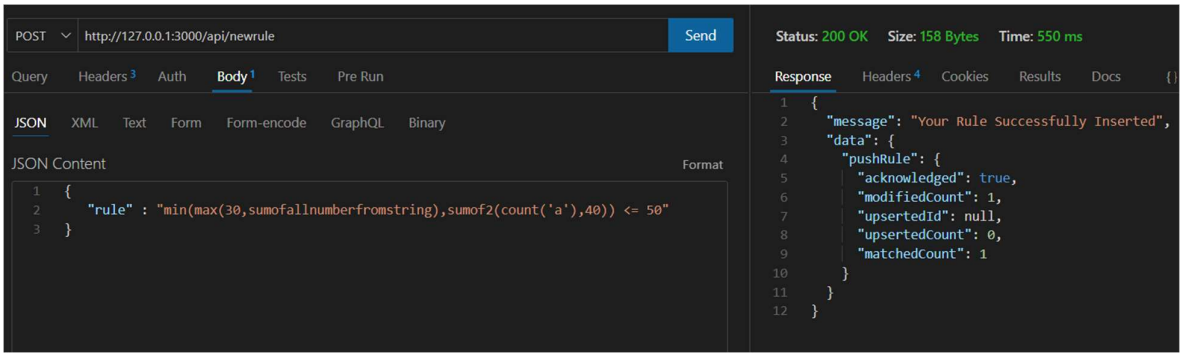
Images



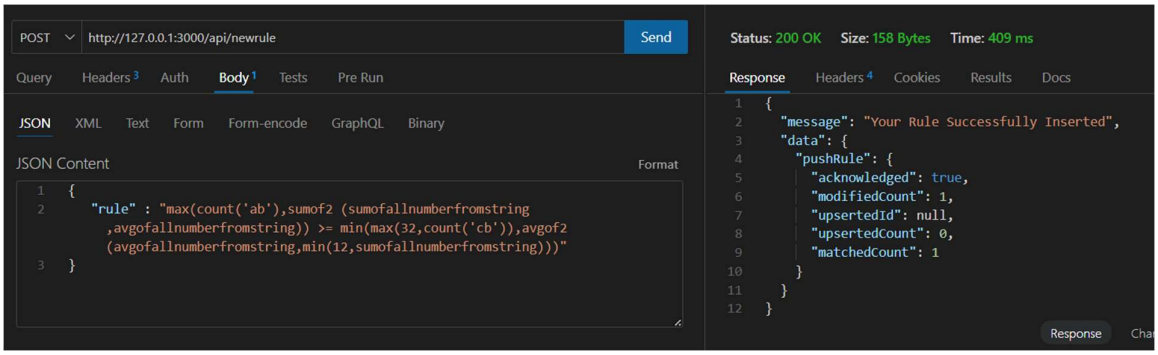
Creating a new User



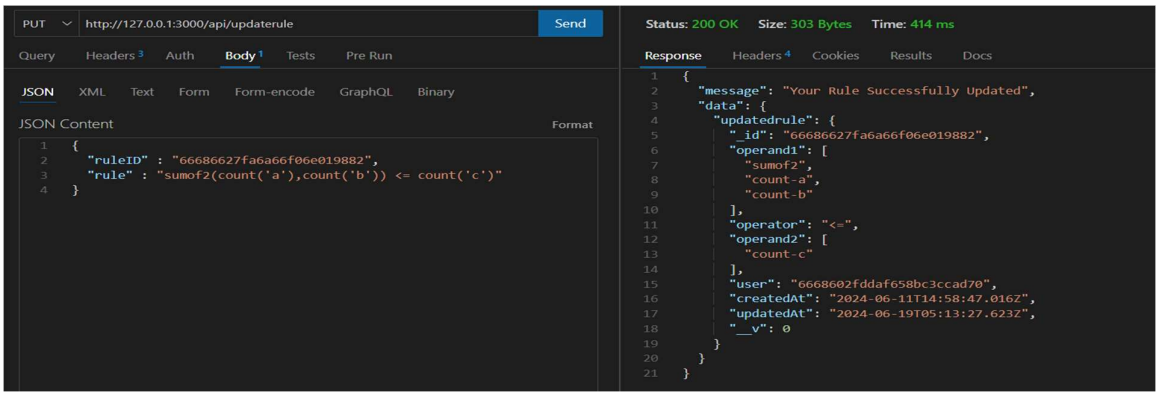
Login a User



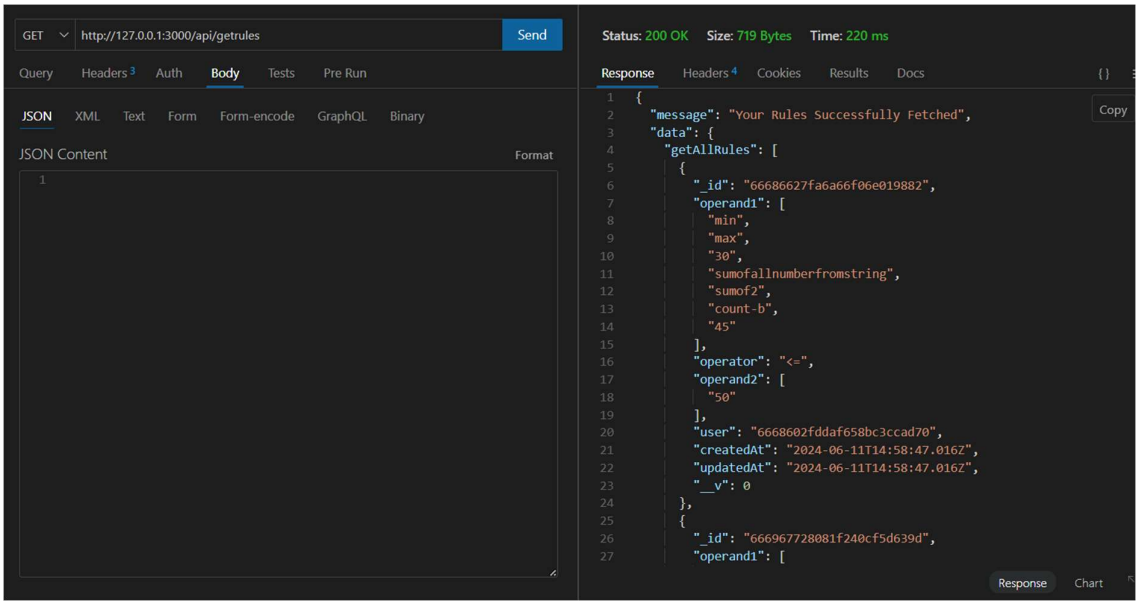
Creating a New Rule



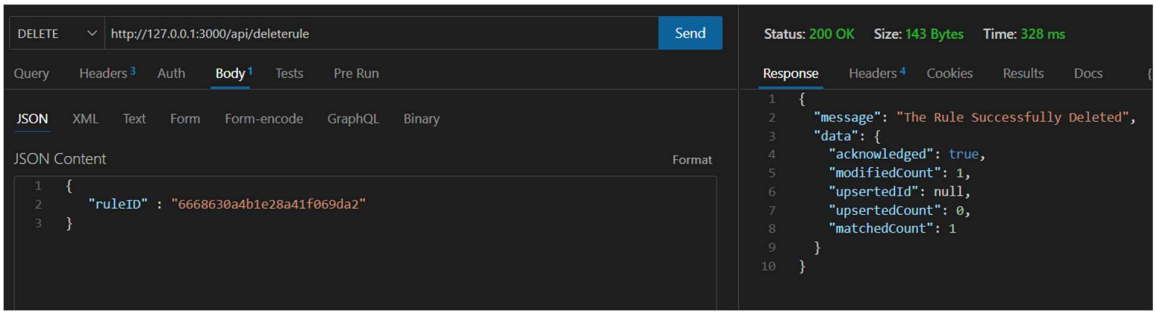
Creating anathor New Rule



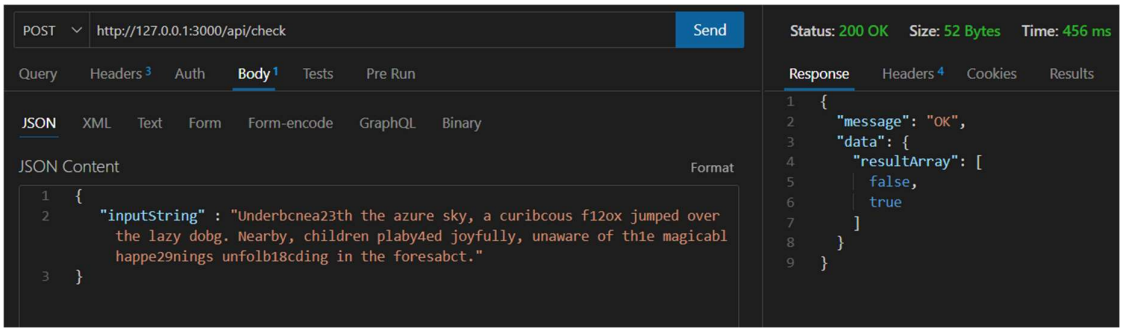
Update a Rule



Get all the stored Rules for a user



Delete a Rule



Applying rules on inputstring and getting result

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	71.31	68.29	86.66	74.05	
src	100	100	100	100	
createApp.js	100	100	100	100	
src/constants	100	100	100	100	
constant.js	100	100	100	100	
src/controllers	54.46	50.98	63.63	57.61	
controller.js	54.46	50.98	63.63	57.61	186-290,309-311,327-328,331-332,335-338
src/models	100	100	100	100	
rule.js	100	100	100	100	
user.js	100	100	100	100	
src/routes	100	100	100	100	
route.js	100	100	100	100	
src/services	98.26	96.66	100	100	
service.js	98.26	96.66	100	100	87,188

Test Suites: 3 passed, 3 total
Tests: 75 passed, 75 total
Snapshots: 0 total
Time: 6.246 s

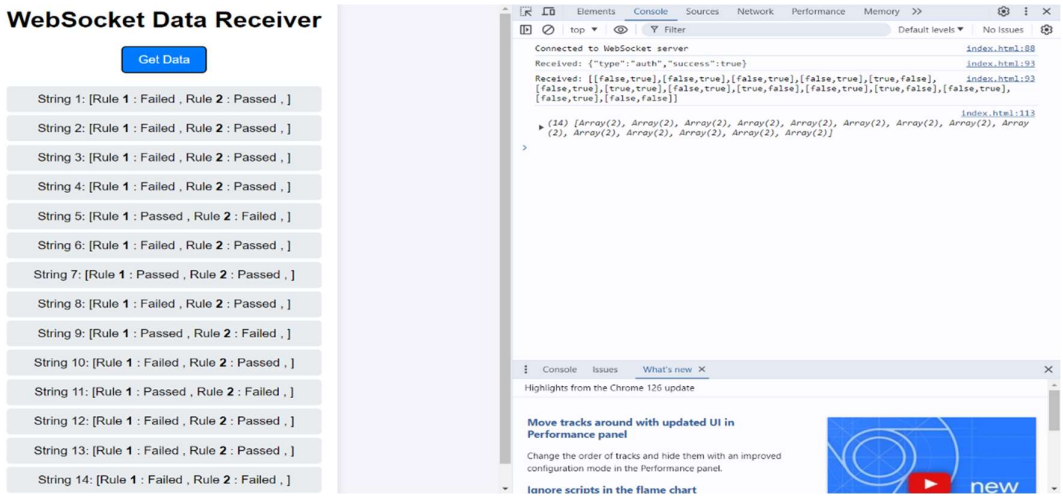
Unit Test/Integration Test/Code coverage

A	B	C	D	E	F	G	H	I	J	K
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	650	11075	381	14409	2736.04	0.00%	42.14485	9.18	20.5	223
TOTAL	650	11075	381	14409	2736.04	0.00%	42.14485	9.18	20.5	223

Matrices obtained during Load Testing (Ramp up Period-1sec)

A	B	C	D	E	F	G	H	I	J	K
Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec	Avg. Bytes
HTTP Request	650	11075	381	14409	2736.04	0.00%	42.14485	9.18	20.5	223
TOTAL	650	11075	381	14409	2736.04	0.00%	42.14485	9.18	20.5	223

Matrices obtained during Load Testing (Ramp up Period-5sec)



WebSocket Data streaming