EmailService Documentation

The custom EmailService that is being used in this device is an Express API that sends mails utilizing different providers, having advanced capabilities like retry logic, circuit breaking, rate limiting, and idempotency. This API accepts email sending requests and provides feedback about the state of each mail and provider.

Installation

Ensure you have Node.js and npm installed. Then, include the necessary modules in your project.

```
npm install express body-parser
```

Files Overview

• index.js: The entry point of the application that sets up an Express server and defines the API endpoints for sending emails.

• main.js: Contains the implementation of the EmailService, the email providers (EmailProviderA and EmailProviderB), and the logic for sending emails.

• test.js: A test script to demonstrate and validate the functionality of the EmailService.

EmailService Class

Constructor

```
constructor(providers)
```

• providers: An array of email providers that the service can use to send emails. Each provider should implement a sendEmail method that simulates sending an email.

Properties

• statusMap: A Map to track the status of sent emails for idempotency and status tracking.

• maxRetries: The maximum number of retry attempts for each provider.

• backoffFactor: The base delay in milliseconds for exponential backoff during retries.

• circuitBreakerThreshold: The threshold for circuit breaking, i.e., the number of consecutive failures before a provider is temporarily disabled.

• rateLimit: The maximum number of emails that can be sent in a given interval.

• rateLimitInterval: The time interval in milliseconds for the rate limiting.

Methods

sendEmail(emails, emailStatus, providerStatus)

• emails: An array of email objects to be sent. Each object should contain to, subject, and body properties.

• emailStatus: An array to capture the status of each email after processing.

• providerStatus: An array to capture the status of each provider after processing.

sendEmailAttempt(email): This method adds the emails to a queue and processes them with rate limiting, retry logic, and provider fallback.

sendWithProvider(provider, email): Attempts to send a single email using the available providers with retry logic and exponential backoff. It also checks for circuit breaker conditions.

sendWithProvider(provider, email): Attempts to send an email using a specific provider. Tracks success and failure counts and manages the circuit breaker.

processQueue(emailStatus, providerStatus): Processes the email queue with rate limiting. After processing, it updates emailStatus and providerStatus with the results.

delay(ms): A utility method that creates a delay in execution, used for implementing backoff and rate limiting.

generateUniqueId(email): Generates a unique identifier for an email based on its content. This is used to ensure idempotency.

Usage

Setting Up the Server

In index.js, the Express server is set up with endpoints to send emails.

```
app.post('/send-emails', async (req, res) => {
  const emails = req.body.emails;

  if (!Array.isArray(emails)) {
      return res.status(400).json({ error: 'Invalid input' });
```

```
    }

    await emailService.sendEmail(emails, emailStatus,
        providerStatus);
    res.status(200).json({
        emailStatus: emailStatus,
        providerStatus: providerStatus
    });
});
```

Running the Server

Run the server with the following command:

```
node index.js
```

The server will start, and you can send POST requests to /send-emails with an array of emails to trigger the service.

Testing

In test.js, a set of test emails are defined and sent through the EmailService. This script helps verify the functionality of the service.

```
emailService.sendEmail(emails, emailStatus, providerStatus);
```

Run the test with:

```
node test.js
```

Extending the Service

Adding New Providers

To add a new provider, create a class that implements a sendEmail method and pass an instance of it to the EmailService constructor.

```
class EmailProviderC {
    async sendEmail(email) {
        // Implement provider-specific email sending logic here
        return true; // or false based on success/failure
    }
}

const emailProviderC = new EmailProviderC();
const emailService = new EmailService([emailProviderA,
        emailProviderB, emailProviderC]);
```

Adjusting Configurations

You can adjust the retry logic, rate limiting, and circuit breaker settings by modifying the properties in the EmailService constructor.

```
this.maxRetries = 5; // Increase the number of retries
this.rateLimit = 10; // Increase the rate limit
this.circuitBreakerThreshold = 7; // Increase the failure
        threshold for the circuit breaker
```

Conclusion

The EmailService gives email sending a complete solution through multiple providers with failures management and high reliability.