Resilient Email Service - README

Features

- Retry Logic: Configurable retry mechanism with exponential backoff and max attempt limits.
- Circuit Breaker: Avoid repeated failures by pausing attempts to failing providers.
- Rate Limiting: Prevent abuse by limiting requests in a time window.
- Idempotency Support: Prevent duplicate email sends using unique keys.
- Email Queue: Queue infrastructure ready for scaling (processing method stubbed).
- Provider Agnostic: Easily add support for real-world providers like SendGrid, Mailgun, SES, etc.
- Health Checks: Visibility into system and provider health.
- Testing: Built-in test runner with multiple test cases.

Setup

Requirements:

- Node.js (v14 or higher)
- TypeScript (recommended for development)

Installation:

```
git clone <your-repo-url>
cd resilient-email-service
npm install
```

Compile TypeScript (if needed):

```
tsc
```

Usage

Running the Example:

```
node dist/<compiled-file>.js
```

Or directly via ts-node:

```
npx ts-node index.ts
```

Resilient Email Service - README

This runs:

- A mock email sending scenario
- Service statistics and health checks

Running Tests

Run Tests:

```
node dist/<compiled-file>.js
```

You will see output for:

- Basic email sending
- Idempotency enforcement
- Rate limiting behavior

Email Service API

sendEmail(message, idempotencyKey?):

Sends an email using available providers. Supports retries, rate limiting, and idempotency.

getAttemptStatus(attemptId):

Get details of a specific email attempt.

getStats():

Returns internal stats (attempts, rate limiter, circuit breakers, etc.)

healthCheck():

Returns service health including circuit breaker status of providers.

Key Interfaces & Classes

EmailService: Main class managing all providers, rate limiting, retries, and metrics.

EmailProvider: Interface to implement custom providers.

MockEmailProvider: A simulated provider with configurable failure rate & latency.

Resilient Email Service - README

RateLimiter: Simple in-memory token bucket implementation.

CircuitBreaker: Handles automatic open/close of unreliable providers based on failures.

Logger: Simple logging wrapper.

EmailQueue: Stub for async queue processing.

Example EmailMessage Format

```
const email = {
  to: 'user@example.com',
  from: 'noreply@company.com',
  subject: 'Hello!',
  body: 'This is a test email.'
};
```

License

MIT License

Contributions

Feel free to fork and improve:

- Provider integrations
- Persistent queueing
- Structured loggers (e.g., Winston)
- Dashboard UI for monitoring