API Rate Limiter for a Payment Gateway

Your application supports multiple payment methods:

- Credit Card
- UPI
- Net Banking

All payment methods call the same external **Payment Gateway API**.

The payment provider has imposed the following restrictions:

1. The system must maintain a **single API session controller**.
   If multiple session controllers are created, the gateway will invalidate previous sessions.

2. The gateway allows **only 5 API calls per second per session**.
   If more than 5 requests are made within the same second using the same session, the gateway will throw a rate-limit error.

System Requirements

- All payment services must use the **same API session**.
- Only **one API session controller** should exist throughout the application.
- The system simulates high traffic by generating **9 API calls within the same second (3 of each payment type)**.
- The correct implementation **must generate a rate-limit error** because 9 > 5.

If your implementation does not generate an error, the design is incorrect.

```
class PaymentGatewaySession {

    private sessionId: string;
    private requestCount: number = 0;
    private currentSecond: number;

    constructor() {
        this.sessionId = Math.random().toString(36).substring(7);
        this.currentSecond = Math.floor(Date.now() / 1000);
        console.log("New Gateway Session Created:", this.sessionId);
    }

    makeApiCall() {
        const now = Math.floor(Date.now() / 1000);

        if (now !== this.currentSecond) {
            this.currentSecond = now;
            this.requestCount = 0;
        }

        this.requestCount++;

        if (this.requestCount > 5) {
            throw new Error("Rate limit exceeded! More than 5 requests
per second.");
        }

        console.log("API Call Allowed. Count:", this.requestCount);
    }

    getSessionId(): string {
        return this.sessionId;
    }
}

class CreditCardPayment {
    process(amount: number) {
        const session = new PaymentGatewaySession();
        session.makeApiCall();
        console.log("Credit Card Payment of", amount,
                    "using session", session.getSessionId());
    }
}

class UPIPayment {
```

```
    process(amount: number) {
        const session = new PaymentGatewaySession();
        session.makeApiCall();
        console.log("UPI Payment of", amount,
                    "using session", session.getSessionId());
    }
}

class NetBankingPayment {
    process(amount: number) {
        const session = new PaymentGatewaySession();
        session.makeApiCall();
        console.log("NetBanking Payment of", amount,
                    "using session", session.getSessionId());
    }
}

const cc = new CreditCardPayment();
const upi = new UPIPayment();
const net = new NetBankingPayment();

for (let i = 0; i < 3; i++) {
    cc.process(1000 + i);
    upi.process(500 + i);
    net.process(2000 + i);
}
```