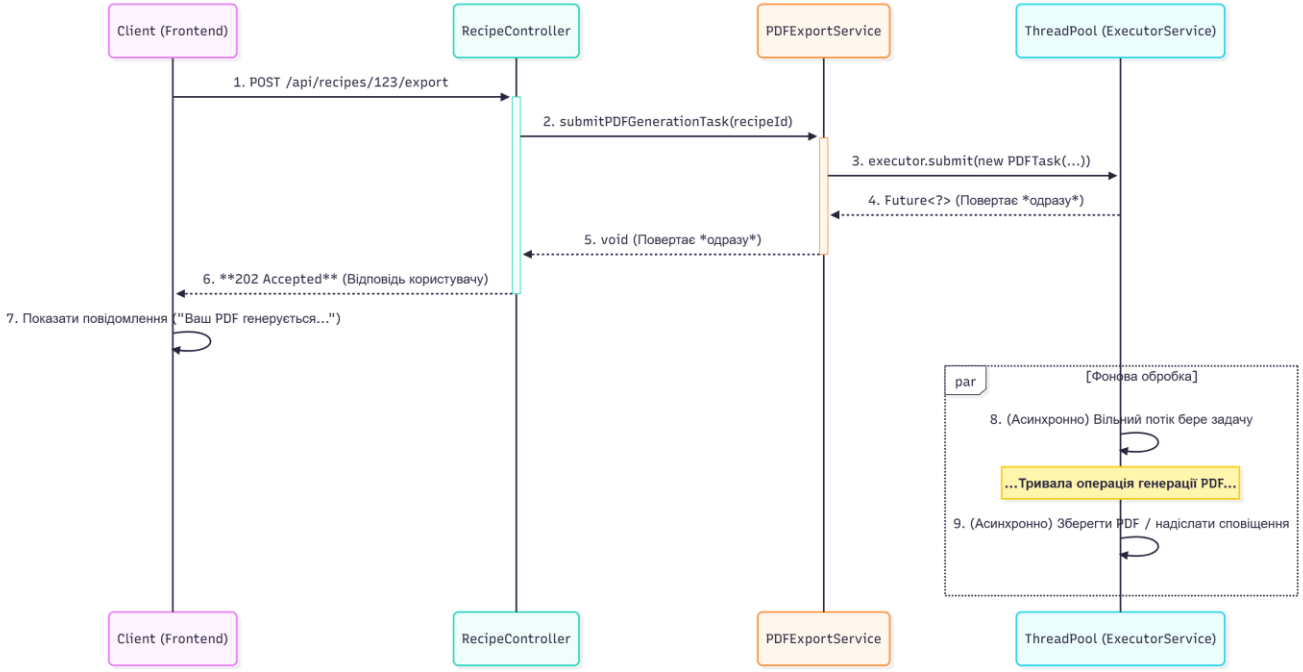
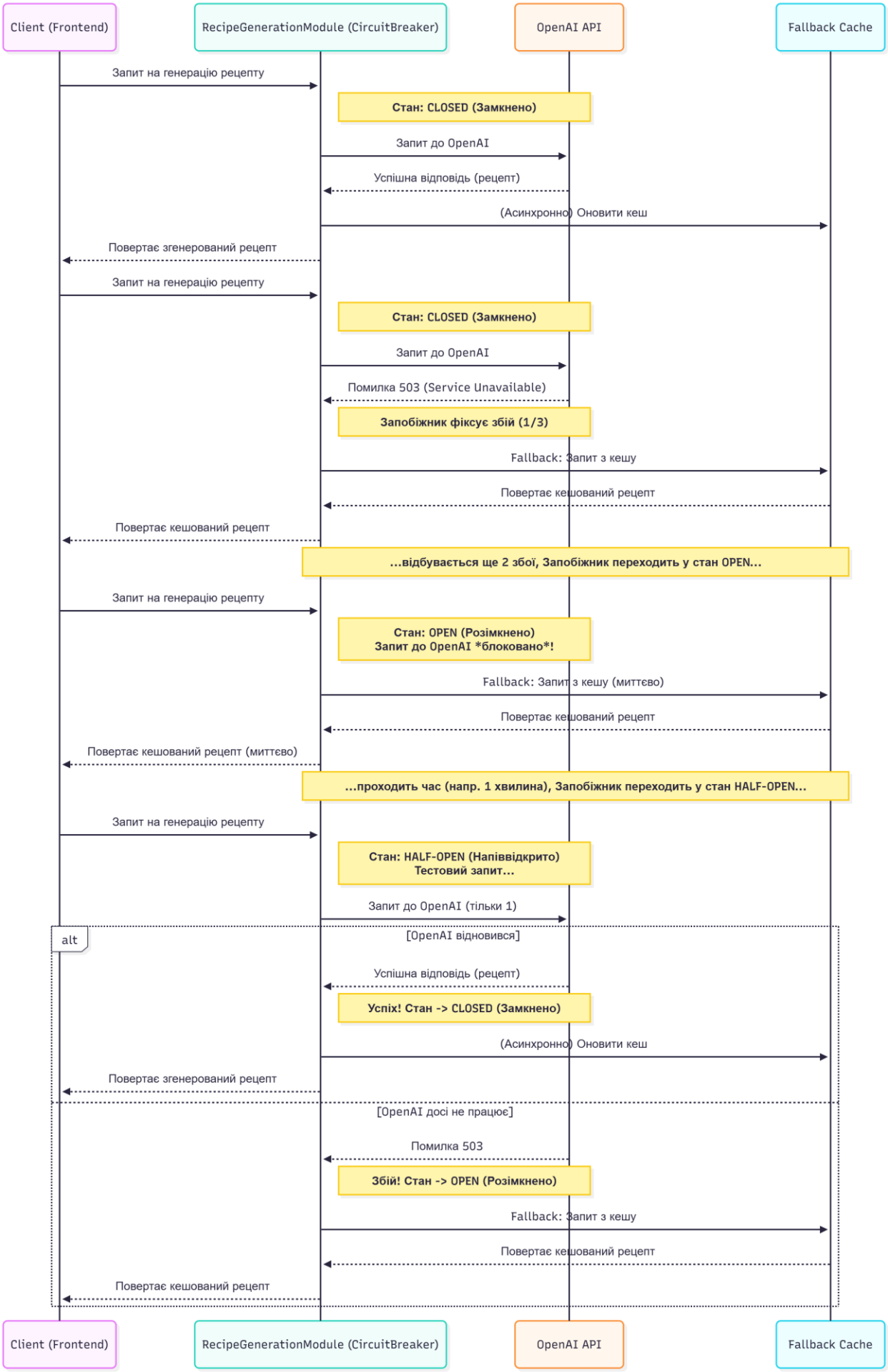
**Concurency Patterns usage**

1. Обробка запитів на експорт рецептів у PDF(Thread pool)
   Генерація PDF може зайняти деякий час і блокувати основний потік обробки HTTP-запитів, призводячи до "зависання" інтерфейсу користувача. Для вирішення цієї проблеми використовується асинхронна обробка з використанням пулу потоків (`Thread Pool`).
   Коли Client надсилає запит на експорт, RecipeController приймає його. Він викликає метод у PDFExportService, який запускає завдання в окремому потоці з ThreadPool. RecipeController не чекає на завершення генерації PDF, а миттєво повертає Client відповідь 202 Accepted.



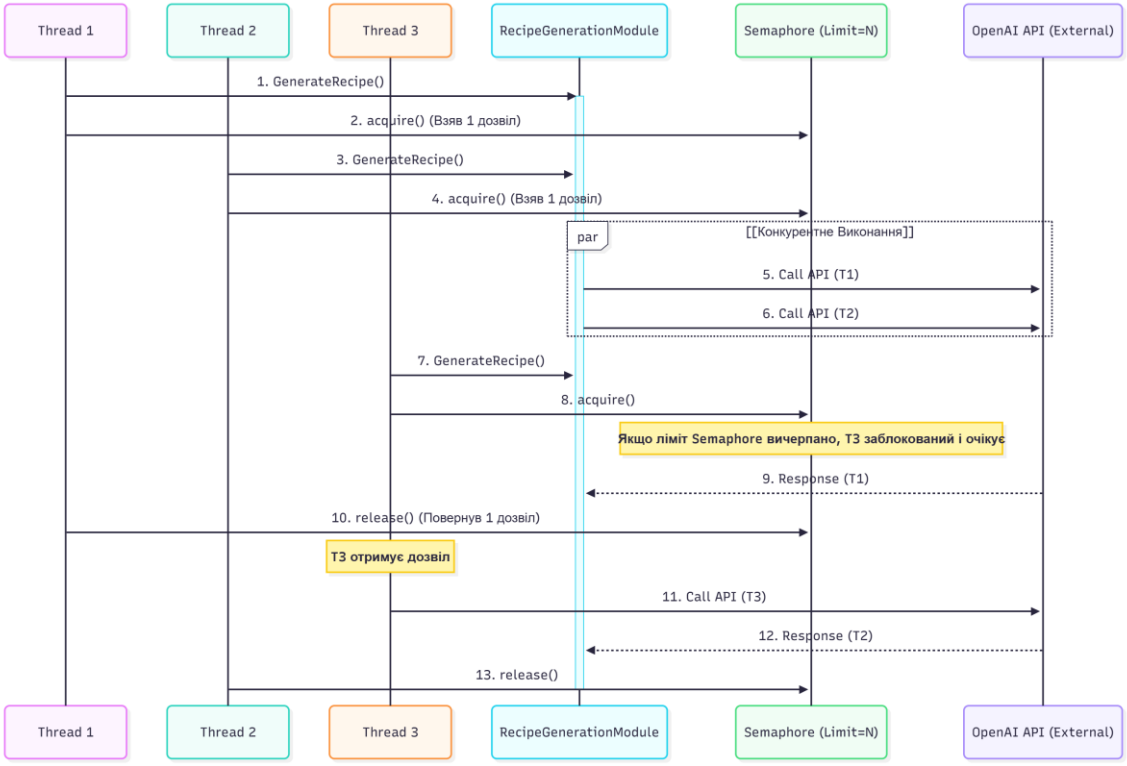

2. Захист AI-генерації в разі збоїв
   Цей потік демонструє використання патерну Circuit Breaker у RecipeGenerationModule для забезпечення надійності та захисту системи від збоїв у зовнішньому OpenAI API. Circuit Breaker має три стани:
   - CLOSED: У нормальному стані Circuit Breaker замкнений, і запити від Client вільно проходять до OpenAI API. Успішні відповіді повертаються клієнту та кешуються.
   - OPEN: Після досягнення порогової кількості збоїв Circuit Breaker переходить у стан OPEN. У цьому стані всі наступні запити до RecipeGenerationModule блокуються і не відправляються до OpenAI API. Замість цього модуль активує fallback-механізм, звертаючись до кешу за останнім збереженим релевантним рецептом. Це захищає систему від перевантаження та забезпечує миттєву відповідь користувачу, навіть коли OpenAI API не працює.
   - HALF-OPEN: Через визначений проміжок часу (наприклад, 1 хвилину) Circuit Breaker переходить у стан HALF-OPEN. Він дозволяє одному "тестовому" запиту пройти до OpenAI API. Якщо цей запит успішний, Circuit Breaker "закривається" і повертається до нормальної роботи. Якщо ж запит знову зазнає невдачі, Circuit Breaker повертається у стан OPEN, продовжуючи захищати систему.


Client (Frontend)
RecipeGenerationModule (CircuitBreaker)
OpenAI API
Fallback Cache
Запит на генерацію рецепту
Стан: CLOSED (Замкнено)
Запит до OpenAI
Успішна відповідь (рецепт)
(Асинхронно) Оновити кеш
Повертає згенерований рецепт
Запит на генерацію рецепту
Стан: CLOSED (Замкнено)
Запит до OpenAI
Помилка 503 (Service Unavailable)
Запобіжник фіксує збій (1/3)
Fallback: Запит з кешу
Повертає кешований рецепт
Повертає кешований рецепт
...відбувається ще 2 збої, Запобіжник переходить у стан OPEN...
Запит на генерацію рецепту
Стан: OPEN (Розімкнено)
Запит до OpenAI *блоковано*!
Fallback: Запит з кешу (миттєво)
Повертає кешований рецепт
Повертає кешований рецепт (миттєво)
...проходить час (напр. 1 хвилина), Запобіжник переходить у стан HALF-OPEN...
Запит на генерацію рецепту
Стан: HALF-OPEN (Напіввідкрито)
Тестовий запит...
Запит до OpenAI (тільки 1)
alt
[OpenAI відновився]
Успішна відповідь (рецепт)
Успіх! Стан -> CLOSED (Замкнено)
(Асинхронно) Оновити кеш
Повертає згенерований рецепт
[OpenAI досі не працює]
Помилка 503
Збій! Стан -> OPEN (Розімкнено)
Fallback: Запит з кешу
Повертає кешований рецепт
Повертає кешований рецепт
Client (Frontend)
RecipeGenerationModule (CircuitBreaker)
OpenAI API
Fallback Cache

3. Керований доступ до OpenAI API (Semaphore)

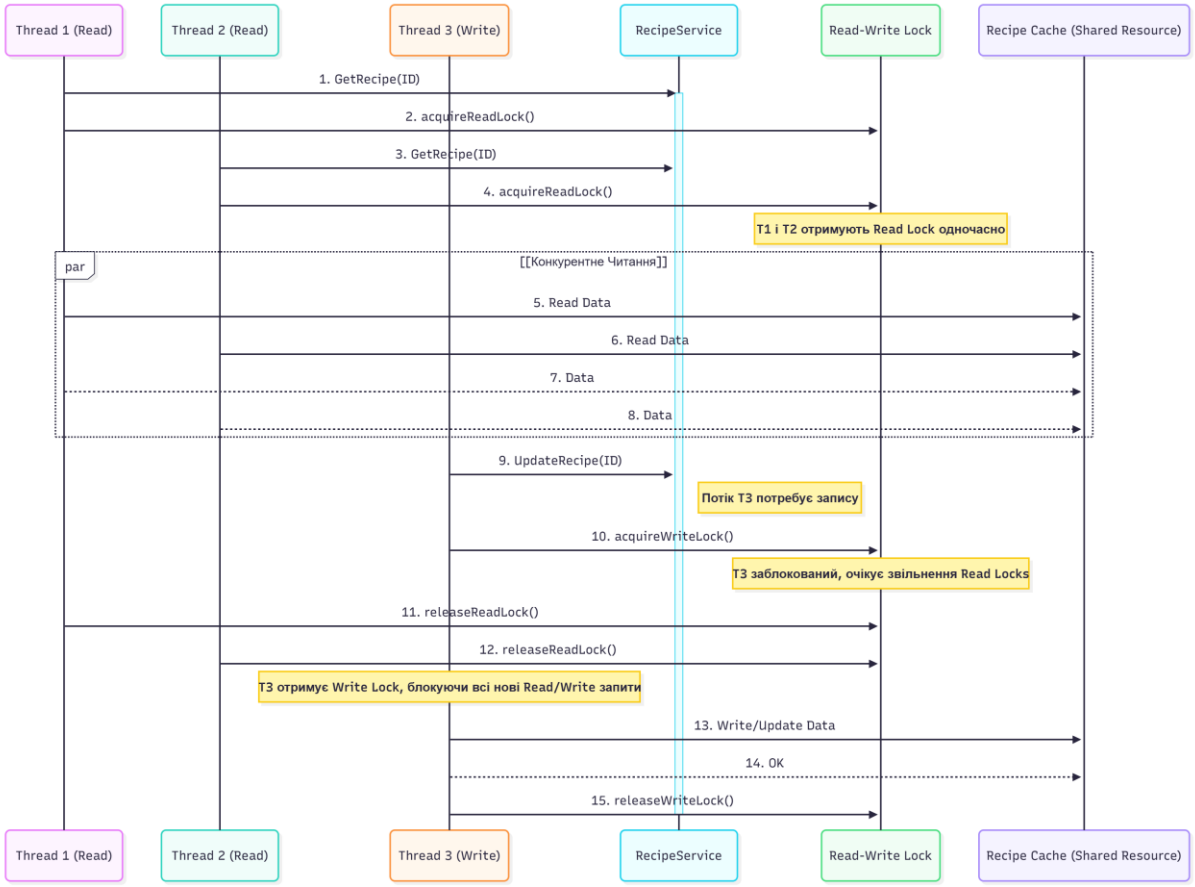
   Суть застосування: Патерн використовується для обмеження кількості конкуруючих потоків, які можуть одночасно виконувати певну операцію. Це критично для взаємодії з дорогими зовнішніми сервісами OpenAI API, які мають суворі ліміти частоти запитів (Rate Limits). Semaphore запобігає перевищенню цих лімітів, навіть якщо внутрішній Thread Pool має багато вільних потоків.

   - Обмеження ресурсів: Semaphore ініціалізується з фіксованою кількістю "дозволів", що відповідає безпечному ліміту викликів до OpenAI API.
   - Контрольований доступ: Кожен потік, перш ніж відправити запит на генерацію рецепту до OpenAI, повинен викликати acquire() на Semaphore (взяти дозвіл).
   - Блокування: Якщо всі дозволи вже використовуються (тобто N одночасних запитів вже обробляються), наступний потік буде заблокований і чекатиме, доки один із активних потоків не звільнить дозвіл (release()).
   - Значення для Конкурентності: Патерн забезпечує керовану конкуренцію. Він запобігає перевантаженню зовнішнього API, що могло б призвести до помилок 429 Too Many Requests і тимчасових блокувань, забезпечуючи стабільний час відповіді для користувачів.

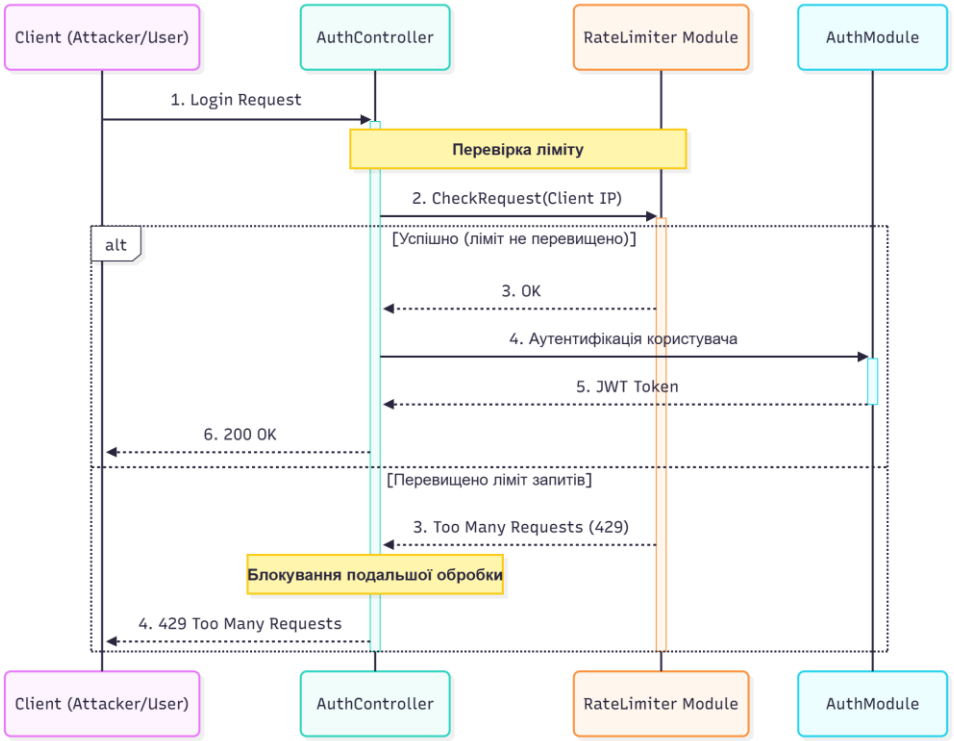



4. Захист спільного кешу (Read-Write Lock)
   Це патерн синхронізації, призначений для оптимізації доступу до спільних даних, коли операції читання відбуваються набагато частіше, ніж операції запису. У Вашому проєкті він ідеально підходить для захисту Recipe Cache (де зберігаються вже згенеровані рецепти для механізму Circuit Breaker).
   - Конкурентне читання: Патерн дозволяє багатьом потокам (наприклад, одночасним запитам користувачів) конкурентно отримувати Read Lock і читати дані з кешу. Це значно підвищує продуктивність, оскільки потоки не блокують один одного.

- Ексклюзивний запис: Коли модуль (наприклад, після успішної генерації нового рецепту) потребує запису або оновлення кешу, він має отримати Write Lock. Цей Write Lock є монопольним — він блокує всі інші потоки, як на читання, так і на запис, гарантуючи, що оновлення відбувається атомарно і цілісність кешу не порушується.
- Значення для Конкурентності: Вирішує проблему стану гонки (race condition) під час запису, але водночас мінімізує блокування під час поширених операцій читання.




5. Обробка одночасних входів (Rate Limiter)
   Патерн для управління конкуруючим доступом до певного ресурсу з метою безпеки та стабільності. Він обмежує кількість запитів, що надходять від одного джерела (наприклад, IP-адреси або ID користувача) за певний проміжок часу.
   Це критично для захисту ендпоінтів аутентифікації (вхід/реєстрація) від атак типу Brute-Force або DoS. AuthController використовує RateLimiter Module для перевірки, чи не перевищує потік встановлений ліміт (наприклад, 3 спроби входу за хвилину). Це захищає AuthService та базу даних від надмірного навантаження.
   Значення для Конкурентності: Патерн забезпечує контрольовану конкуренцію. Він запобігає виснаженню обчислювальних ресурсів моноліту, гарантуючи, що велика кількість одночасних (конкурентних) запитів від зловмисника не вплине на доступність системи для легітимних користувачів.


Client (Attacker/User)
AuthController
RateLimiter Module
AuthModule
1. Login Request
Перевірка ліміту
2. CheckRequest(Client IP)
alt
[Успішно (ліміт не перевищено)]
3. OK
4. Аутентифікація користувача
5. JWT Token
6. 200 OK
[Перевищено ліміт запитів]
3. Too Many Requests (429)
Блокування подальшої обробки
4. 429 Too Many Requests
Client (Attacker/User)
AuthController
RateLimiter Module
AuthModule