Case Study 2: User Profile Service with Caching 💾

The Scenario: A social media platform, "Connectly," has a **User Profile Service** that fetches user data from a database. Another service, the **Timeline Service**, constantly calls the User Profile Service to fetch data for displaying posts. The database is a single point of failure and can sometimes become slow or unreachable. If the database goes down, the User Profile Service will fail, causing the Timeline Service to fail, and so on.

The Solution with a Circuit Breaker and Fallback Cache: Connectly implements a circuit breaker to protect against database failures.

- **Closed State**: The circuit breaker is **closed**, and the User Profile Service fetches data directly from the database.
- **Threshold Trigger**: The database experiences a period of high load, causing requests to the User Profile Service to timeout. The circuit breaker detects this based on its configured threshold (e.g., **90%** of calls failing).
- **Open State**: The circuit breaker opens, preventing any more requests from hitting the database.
- **Fallback with a Cache**: In the `fallbackMethod`, the User Profile Service does not return a generic error. Instead, it attempts to retrieve the requested user data from a **local cache** (e.g., Redis or an in-memory cache).
    - If the data is found in the cache, it's returned to the Timeline Service. This allows the Timeline Service to continue displaying timelines with slightly outdated but still usable data.
    - If the data is not in the cache, it returns a generic fallback response.
- **Half-Open State**: After a set time, the circuit breaker transitions to the **half-open state**, allowing a few requests to test the database connection.
- **Recovery**: If the database has recovered, the test requests succeed, the circuit breaker closes, and the User Profile Service returns to fetching fresh data from the database.