Documentation

**A} Overview**

**Cachetron** is a **unified caching abstraction layer** designed to simplify the management of multiple caching backends in your applications. It enables developers to interact with different caching systems, such as **Redis** and **Memcached**, through a single, consistent interface, eliminating the need to write backend-specific code.

**Key Features**

* **Pluggable Multi-Backend Support:** Easily switch between caching backends without modifying your application logic.
* **Unified Caching Interface:** One consistent API for all supported caching systems.
* **Runtime Backend Switching:** Change the caching backend dynamically during runtime.
* **ML-Based TTL Tuning:** Automatically optimize cache expiration using machine learning predictions.
* **Framework Complexity Abstraction:** Simplifies integration by handling the underlying complexities of different caching frameworks.
* **Extensible & Robust Architecture:** Built for future expansion and high reliability.

**B} Installation**

Follow these steps to install and set up **Cachetron**:

**1. Install Cachetron**

Install Cachetron using **npm**:

npm install -g cachetron

* The -g flag installs Cachetron globally so you can run its commands from anywhere.
* Alternatively, you can install it locally in your project if preferred:

npm install cachetron

**2. Initialize Cachetron**

Set up Cachetron in your project:

Cachetron init

* This command creates the necessary configuration files and prepares your project to use Cachetron.

**3. Launch the Dashboard**

Start the Cachetron dashboard:

Cachetron dashboard

* Opens the dashboard in your default browser.
* Use the dashboard to monitor cache usage, backends, and ML-based TTL settings in real-time.

**C} Usage / Getting Started**

* Show **basic examples** of initializing Cachetron, setting/getting cache, switching backends, etc.
* Include **code snippets** for clarity.

**Example:**

import Cachetron from 'cachetron';

// Initialize Cachetron

const cache = new Cachetron();

// Set a value in the cache

cache.set('user\_123', { name: 'Alice', age: 25 });

// Get a cached value

const user = cache.get('user\_123');

console.log(user); // { name: 'Alice', age: 25 }

// Delete a cached value

cache.delete('user\_123');

**2. Switching Backends**

Explain how to dynamically change the caching backend:

// Switch from Redis to Memcached at runtime

cache.switchBackend('memcached');

**3. Advanced Features**

* **ML-Based TTL Tuning:** Automatically adjust cache expiration.
* **Monitoring via Dashboard:** Show current keys, TTLs, and backend usage

**D} Command line interface**