A Project Synopsis

On

Sustainable Caching Integration for Monolith API Framework

For the Degree of
Bachelor of Technology

In
Computer Science and Engineering
By

Pranav Dilip Hingankar
2020BTECS00040

Under the Guidance of
Mr. Raunak Mali
Mr. S. D. Pujari

Department of Computer Science and Engineering
Walchand College of Engineering, Sangli
(Government Aided Autonomous Institute)
(AY 2023-24)

Sustainable Caching Integration for Monolith API Framework

Project Domain

Web Development

1. Introduction

The project, titled "Sustainable Caching Integration for Monolith API Framework," addresses potential performance and server load issues within the existing ACD API framework. The primary focus is on implementing sustainable caching logic to optimize overall system performance. Caching involves storing and retrieving data more efficiently, aiming to enhance the efficiency and scalability of the Monolith API framework.

2. Problem Statement

To introduce sustainable caching to alleviate strain, particularly for frequently accessed APIs, aiming to enhance overall system performance.

2.1 Objectives

1. Implement efficient caching in the Monolith API.
2. Create POCs for diverse caching strategies.
3. Align caching mechanisms with architectural goals.
4. Evaluate caching solutions' performance and scalability.

3. Proposed Approach/Methodology

The project's methodology involves a systematic approach to address potential performance and server load issues within the Monolith API framework through sustainable caching integration. Here's an overview of the methodology:

1. Problem Identification and Analysis:

   Identify performance bottlenecks and server load issues within the Monolith API framework. Analyze usage patterns, system metrics, and API request/response dynamics to pinpoint areas for improvement.

2. Research and Exploration:

Conduct extensive research on caching techniques, best practices, and existing solutions. Explore various caching architectures, algorithms, and technologies suitable integration the Monolith API framework.

3. Proof of Concept Development:

Design and develop Proof of Concepts (POCs) to experiment with different caching strategies. Implement prototype solutions to test the feasibility and effectiveness of caching mechanisms in improving system performance.

4. Architectural Input and Collaboration:

Collaborate with architects and stakeholders to gather input and insights into the overall design and goals of the Monolith API framework. Incorporate architectural recommendations to ensure alignment with the framework's structure and requirements.

5. Caching Implementation:

Implement caching logic within the Monolith API framework to store and retrieve frequently accessed data efficiently. Develop cache management components responsible for overseeing caching operations and coordinating with the decision engine.

6. Decision Engine Development:

Design and develop a decision engine component to analyze usage patterns, system metrics, and real-time data for dynamic caching decisions.
Implement algorithms and strategies to determine optimal caching policies based on current conditions.

7. Feasibility Testing:

Conduct comprehensive feasibility testing to evaluate the performance, reliability, and scalability of the proposed caching solutions. Test different caching configurations and options under realistic conditions to assess their effectiveness in reducing server load and improving system efficiency.

8. Logging and Monitoring Integration:

Integrate logging and monitoring capabilities to capture relevant system metrics, usage patterns, and caching performance data. Use logging and monitoring data to support decision-making processes in the decision engine and monitor the overall performance of the caching system.

9. Evaluation and Analysis:

Evaluate the results of the caching integration, including performance improvements, reduced server load, and optimized resource utilization. Analyze simulation results, including cache hit rates and comparative analysis before and after implementation, to measure the impact of caching on system performance.

4. Deliverables

To deliver enhanced system performance, reduced server load, optimized resource utilization, and increased scalability and flexibility.

5. Software Specification

- C#
- Visual Studio
- Redis Cache
- Microsoft SQL Server
- Jira

Mr. S. D. Pujari	**Mr. Raunak Mali**	**Dr. Medha A. Shah**
Guide	Tech Lead, NICE	Head