Food Management System

Final Sprint

Roshini Seelamsetty, Nida Tazeen

Purdue University

ACS 560: Software Engineering

Matthew Parker

9th December 2024

Final Sprint

This document provides an overview of the final sprint for the Food Management System project. The sprint focuses on key performance improvements, modular UI updates, and significant bug fixes. Our team also implemented a robust data caching strategy to optimize system performance.

For the final sprint we worked on these–

#### **Data Caching Integration**

* **Implemented ConcurrentMapCacheManager:**

Used this class to enable caching of frequently accessed data. By caching common requests such as customer or order details, we significantly reduced the database load and improved data retrieval speed.

* **Performance Improvements**:

By caching data in memory, API response times were greatly reduced. This caching mechanism is particularly effective for data that is frequently queried or remains static for a certain period, leading to improved system performance.

* **Caching Strategy**:

Caching is especially useful when the same data is requested multiple times. Rather than querying the database for every request, we store the data in the cache and retrieve it quickly for subsequent requests.

* **Custom CacheManager and LoggingCache**:

A custom CacheManager was implemented to handle the caching process, and a LoggingCache was introduced to log cache operations like hits, misses, evictions, and puts. This helps track cache behavior and performance, identifying any potential issues for debugging and optimization.

* **Cache Operations Logging**: Logs of cache operations (hits, misses, evictions) were implemented to provide transparency into cache behavior. This is essential for monitoring performance and ensuring that the cache reduces the number of database queries as intended.

#### **Code Updates**

#### Modularized the user interface (UI) by creating separate methods for UI components like:

#### createHeader: Separates header creation logic.

#### createInputLayout: Modularizes input fields and controls.

#### createOrderGrid: Modularizes order grid creation.

#### Centralized grid column setup into the configureOrderGrid method for easier maintenance.

#### Enhanced user feedback by integrating Notification.show() for clear, non-intrusive alerts, replacing direct grid text updates.

#### Improved input validation, specifically in the fetchOrderHistory method, ensuring that only valid customer IDs are processed.

#### Improved the layout with better component spacing, button styling, and grid enhancements for better visibility.

#### Fixed the delete method to properly handle foreign key relationships, ensuring that deleting an order also takes care of related customer and restaurant data while adhering to referential integrity rules. This fix prevents potential database errors when deleting records involved in foreign key relationships.

#### **Project Presentation (PPT)**

* Created and delivered a detailed presentation covering:
* Project Overview
* Functional Requirements Met.
* Not implemented features.
* Additional Features Implemented.

**Instructor Requirements Fulfillment**

1. Client/Server Architecture:
   * Implemented using a Spring Boot backend and a user interface built using Vaadin.
2. Server Using Spring Boot:
   * The server was built using Spring Boot with REST APIs for CRUD operations.
3. MySQL Database:
   * Used MySQL for persistent data storage, ensuring structured and relational data management.
4. Server Deployed to Ubuntu (Localhost):
   * The server was deployed and tested on an Ubuntu-based localhost environment.
5. Data Cache System:
   * Implemented using ConcurrentMapCacheManager.
   * The cache updates regularly after CRUD operations, optimizing API response times.
   * Cache logs track hits, misses, evictions, and data updates.

**Conclusion:**

#### Caching Strategy: Implemented in-memory caching to reduce data retrieval times and enhance the overall user experience by delivering faster responses.

#### Code Improvements: Streamlined service components and improved the update logic for better maintainability and performance.

#### **Challenges Faced**

#### Concurrency Issues: Encountered synchronization issues when handling concurrent requests, which required additional attention to ensure thread safety and consistency.

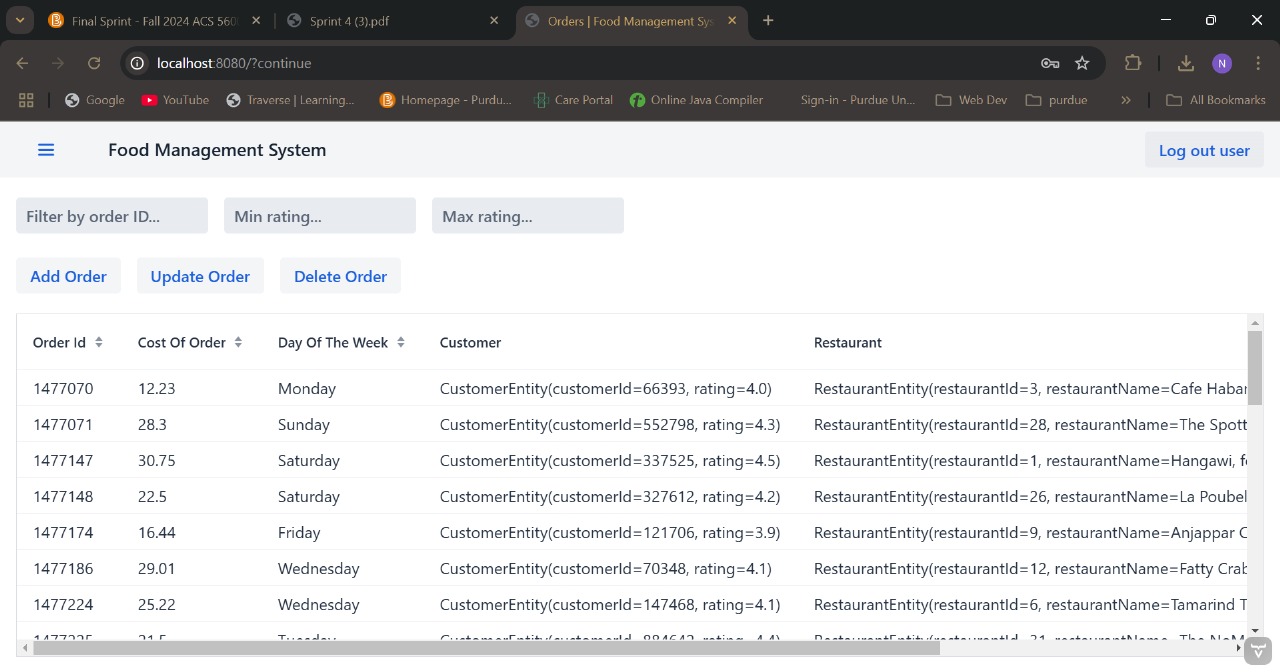
#### Caching Misconfigurations: Initially faced caching misconfigurations that caused data inconsistencies, but these were resolved through further testing and adjustments to the caching setup.

#### **Lessons Learned**

#### Caching Strategies: Effective caching is essential for improving system scalability and performance, especially when dealing with frequently accessed data.

#### Team Communication: Clear and consistent communication within the team is crucial for ensuring smooth project progression and resolving issues promptly.

The following are the screenshots of final sprint –



A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

Git Link:

<https://github.com/tazeenida/Food-Management-System.git>