RV College of Engineering®
Mysore Road, RV Vidyaniketan Post,
Bengaluru - 560059, Karnataka, India

Go, change the world®

DEPARTMENT OF

INFORMATION SCIENCE AND ENGINEERING

Innovative Experiment
Report On

"APIs FOR SERVICE SYNCHRONIZATION"

By

1RV23SIT10 Ravish S H

1RV23SIT07 Surya M Ganesh

1RV23SSE06 Gurukiran P S

Under the Guidance of

Prof. Rashmi R
Assistant Professor
Dept. of ISE
RV College of Engineering®

Course Name: API Development and Integration Lab
Course Code: MIT438L

SEPT 2023 - 24

Table of Contents

Report on Number Printing API with Memory Monitoring and Log Management

INTRODUCTION

This project is designed to implement a distributed Spring Boot application that performs number printing, memory usage monitoring, and server switching in case of memory leaks. The application runs on two separate servers (8080 and 8081), logging developer-centric information (number, timestamp, and memory leak percentage) and user-centric data (printed numbers only) into separate log files. A key feature is that if memory usage exceeds a certain threshold, the application automatically switches between servers without interrupting the printing process. This report discusses the requirements, code implementation, setup instructions, and detailed working of this application.

SOFTWARE REQUIREMENTS WITH VERSION, INSTALLATION PROCEDURES

Following is the list of software requirements specified in order of installation:

Software Name	Version	Installation link	Purpose
Java Development Kit (JDK)	11 or higher	https://www.oracle.com/java/technologies/downloads/#java11	Development environment for Java applications.
Spring Boot	2.6.7 or compatible	https://start.spring.io/	Framework for building RESTful APIs and microservices.
Eclipse IDE	2023-03 or higher		Integrated development environment (IDE) for Java development.
Redis Server	Version 7.0		Enhances application performance for caching
Maven	3.8.6 or higher	https://maven.apache.org/download.cgi	Build tool for managing dependencies and project lifecycle.
Postman	10.1 or higher		Tool for testing and interacting with APIs.

SOURCE CODE LINK (GITHUB):

https://github.com/Ravish7349/api_process_synchronization.git

OVERVIEW OF THE ESSENTIAL ASPECTS OF THE API

The tables below provides a clear and concise overview of the essential aspects of the created APIs, making it easy for users to understand and reuse:

Section	Details
API Overview	**Name: Process Synchronization** **Version: 1.0** **Base URL: http://localhost:8080/start/api**
Authentication	**No authentication required for this version.**
Endpoints	**GET – http:localhost:8080/start/api** **Description: Run server1.** **Parameters: file (multipart/form-data)** **Request Example: http:localhost:8080/start/api** **Response Example: { " Printing paused due to memory leak. Server switched." }**
Error Handling	**400 Bad Request**
Usage Examples	**Example:** **import requests** **response = requests.get('http:localhost:8080/start/api** **, headers={'Authorization': 'Bearer YOUR_API_KEY'})** **print(response.json())**
Additional Resources	**API Documentation: Full Documentation** **Support: Contact Support**

LIST OF APIS WITH ITS PURPOSE

API Name	**Requirements**	**Installation link**	**Purpose**
/api/start	Starts the number printing and memory monitoring process		This API triggers the number printing on the active server. It monitors memory usage and switches servers if memory usage exceeds 80%.
/api/logmerge	Merges the log files from both servers into one file		This API merges the logs generated by both applications into one user-specific log file that contains all the numbers printed from both servers.

DESCRIPTION ABOUT EACH MODULE

1]. ApiController

Handles HTTP requests and exposes the /api/start and /api/logmerge endpoints. It communicates with other services like NumberPrinterService, MemoryMonitorService, and ServerManager to execute core functions.

2]. NumberPrinterService

This service is responsible for printing numbers to the console and saving them to both the developer log (with memory usage details) and the user-specific log (numbers only). It interacts with Redis to retrieve and store the current number across both servers.

3]. MemoryMonitorService

Monitors the memory usage of the running application. It provides memory usage percentages and checks if memory usage exceeds the threshold, triggering a server switch.

4]. ServerManager

Manages the server switching logic. When memory usage exceeds 80%, it switches the application from one server to the other (from 8080 to 8081 and vice versa).

5]. RedisService

Handles interactions with the Redis database, storing and retrieving the current number being printed. It also increments the number as each is printed.

6]. ExcelLogger

Logs developer-centric information (number, timestamp, and memory leak percentage) into an Excel file using the Apache POI library.

7]. LogMergerService

This service is responsible for merging log files from both servers into a single user log file.

IMPLEMENTATION DETAILS WITH TOOLS USED

1. Spring Boot

Overview: Spring Boot is a powerful framework that simplifies the process of building production-ready applications with the Spring framework. It provides a set of conventions and configurations to get applications up and running quickly, reducing the need for boilerplate code and complex configurations. In this project, Spring Boot serves as the core framework for developing and deploying the API that handles number printing, memory monitoring, and server switching.

Implementation Details:

- **Project Setup:** The Spring Boot application is initialized using the @SpringBootApplication annotation, which includes @Configuration, @EnableAutoConfiguration, and @ComponentScan annotations.
- **Controller Layer:** The ApiController class exposes REST endpoints to start number printing and merge logs. It interacts with services like NumberPrinterService, MemoryMonitorService, and ServerManager.
- **Service Layer:** The NumberPrinterService handles number generation and logging, while MemoryMonitorService monitors memory usage and detects leaks. ServerManager manages server switching based on memory leak conditions.

2. Redis

Overview: Redis is an in-memory data structure store used as a database, cache, and message broker. It provides high performance for read and write operations, making it ideal for managing state in distributed systems. In this project, Redis is used to store and manage the current number being printed, ensuring that the number is consistent across multiple server instances.

Implementation Details:

- **Redis Configuration:** The RedisConfig class configures the RedisTemplate bean for interacting with Redis. It sets up the connection factory and value serializer.
- **Service Integration:** The RedisService class interacts with Redis to get and increment the current number, ensuring synchronization between servers.

3. Apache POI

Overview: Apache POI is a Java library used for reading and writing Microsoft Office documents, including Excel spreadsheets. It provides a way to generate, modify, and access Excel files programmatically. In this project, Apache POI is used to log developer-centric details such as numbers, timestamps, and memory leak percentages into an Excel file.

Implementation Details:

- **Excel File Creation:** The ExcelLogger class handles the creation and updating of the Excel file. It initializes the workbook and sheet, adds headers, and appends new rows with log data.
- **Logging Data:** The logData method in ExcelLogger adds a new row for each log entry,

including the number, timestamp, and memory leak percentage.

4. Postman

Overview: Postman is an API development tool used for testing and interacting with APIs. It provides a user-friendly interface for sending HTTP requests, receiving responses, and managing different environments. In this project, Postman is used to test the REST APIs exposed by the Spring Boot application.

Implementation Details:

- API Testing: Postman collections are used to send requests to endpoints like /api/start and /api/logmerge, ensuring that the application behaves as expected under different scenarios.
- Response Validation: The tool helps validate the responses from the server, including the status codes and response data.

WORKING PROCEDURE

The working principle of service synchronization involves ensuring that different components or services operate together seamlessly to achieve a common goal. Here's a detailed outline of how service synchronization typically works:

1. Start the Application:

- Run both applications on ports 8080 and 8081. Both applications will share the same Redis data store for synchronizing the current number being printed.

2. API Request to Start Printing:

- Make a GET request to /api/start using Postman on either server (8080 or 8081).
- The active server begins printing numbers, monitoring memory usage.

3. Memory Leak Detection and Server Switching:

- The MemoryMonitorService checks memory usage during number printing.Preprocessing techniques such as converting to grayscale, resizing, or increasing contrast could be added to this step if required.
- If memory usage exceeds 80%, the application switches to the other server (from 8080 to 8081 or vice versa) and resumes printing from the last number.

4. Log Files:

- Developer-centric logs (number, timestamp, and memory leak) are saved to log.xlsx.
- User-centric logs (numbers only) are saved to app1_user_numbers_log.txt and app2_user_numbers_log.txt.

5. Merge Log Files:

- Use the /api/logmerge endpoint to merge app1_user_numbers_log.txt and app2_user_numbers_log.txt into a single log file containing all numbers.

SCREENSHOTS

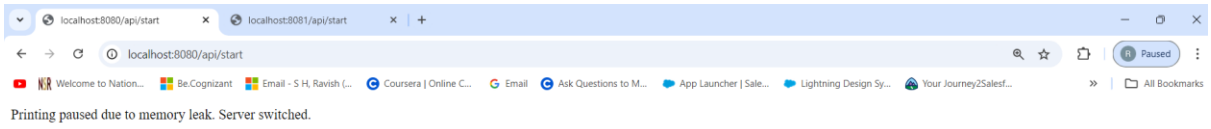

Fig.1 Starting of Server1

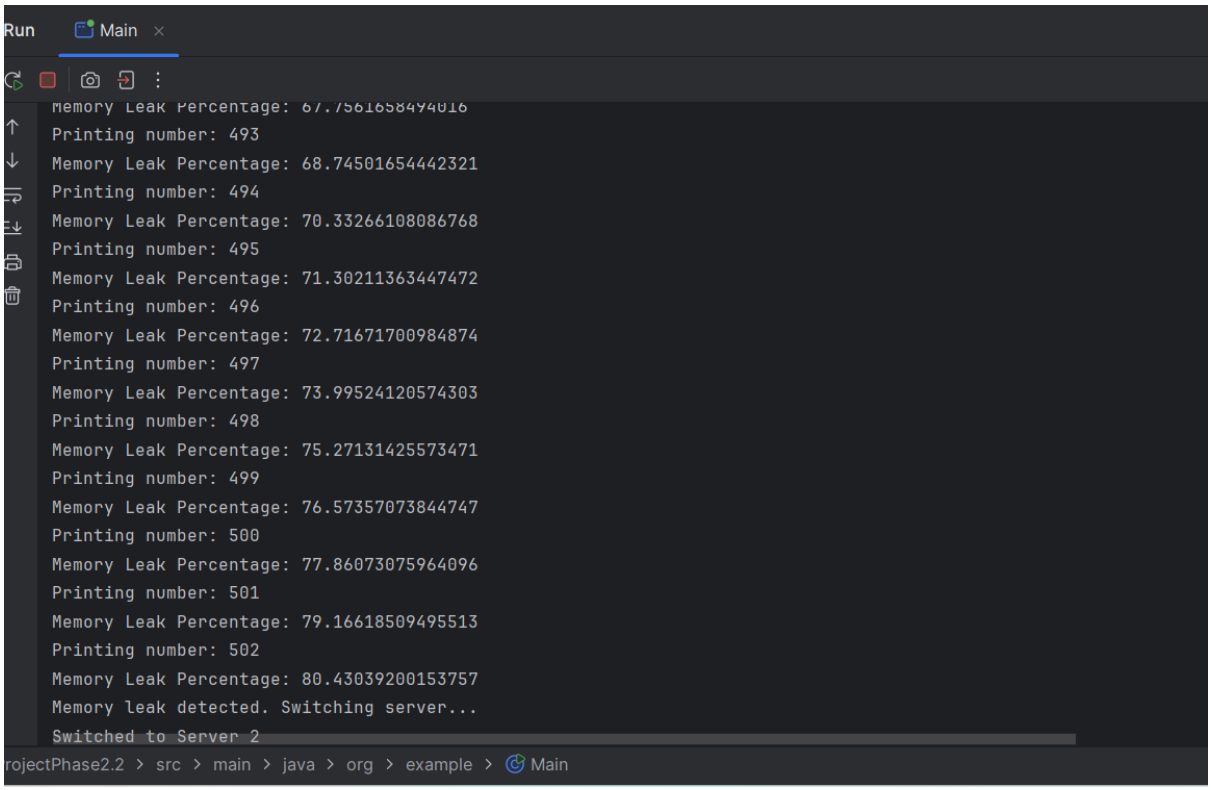

Fig.2 Output after running in server 1

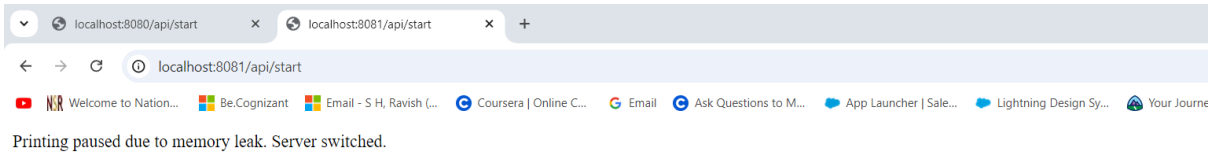

Fig.3 Starting of Server 2

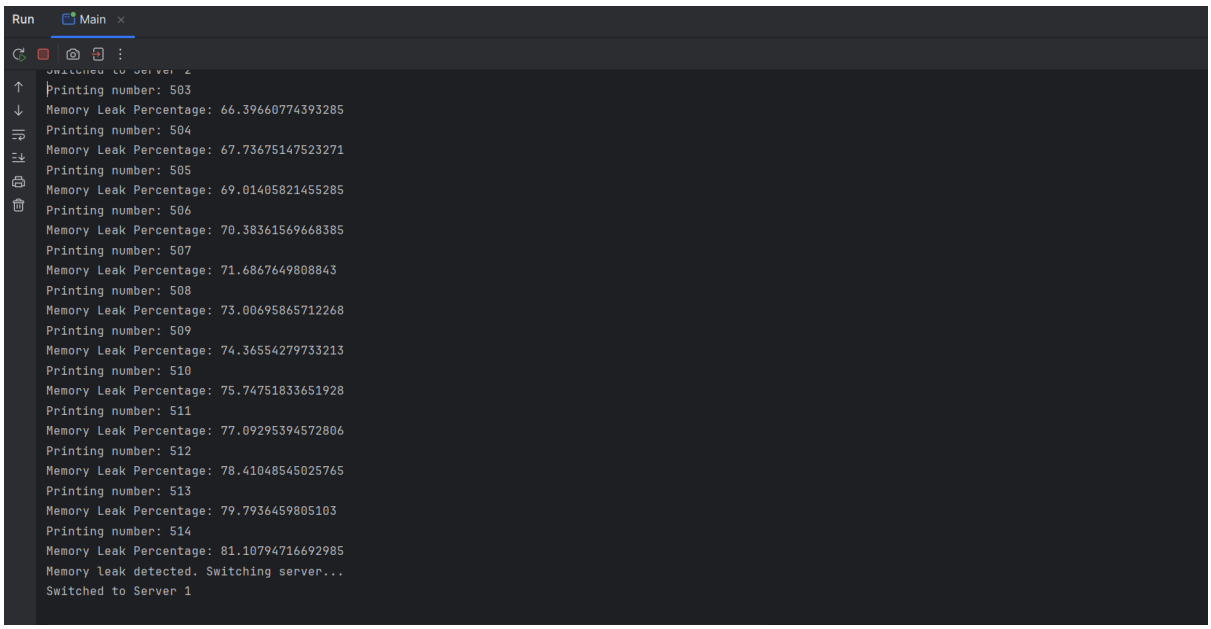

Fig.4 Output after running in server 2

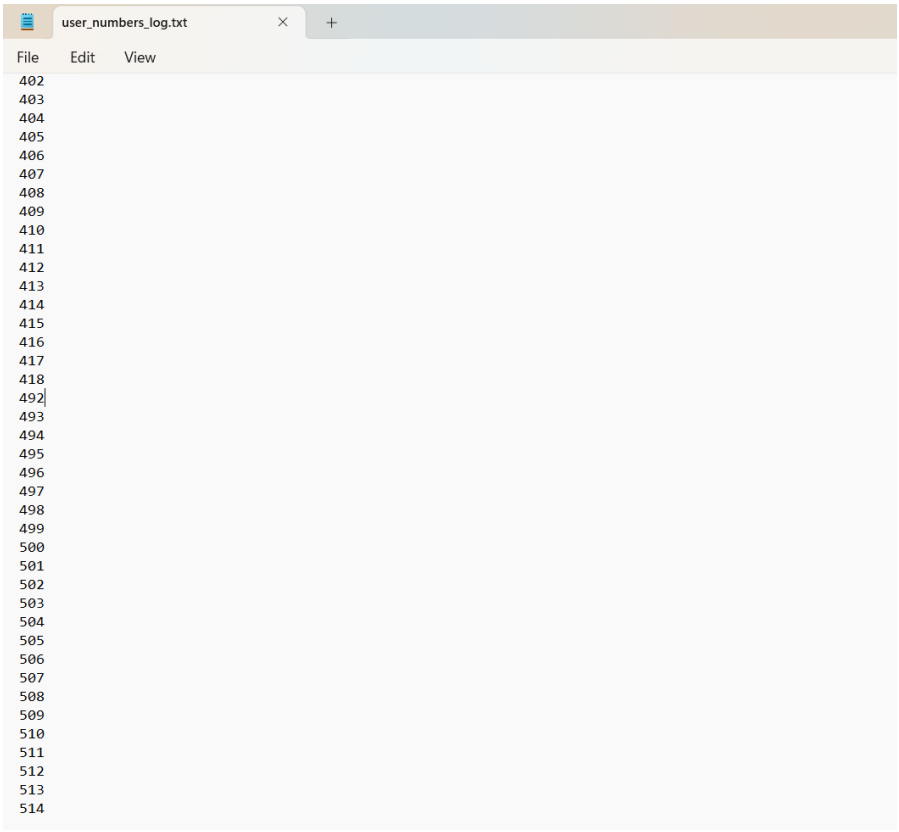

Fig.5 Output of User Interface

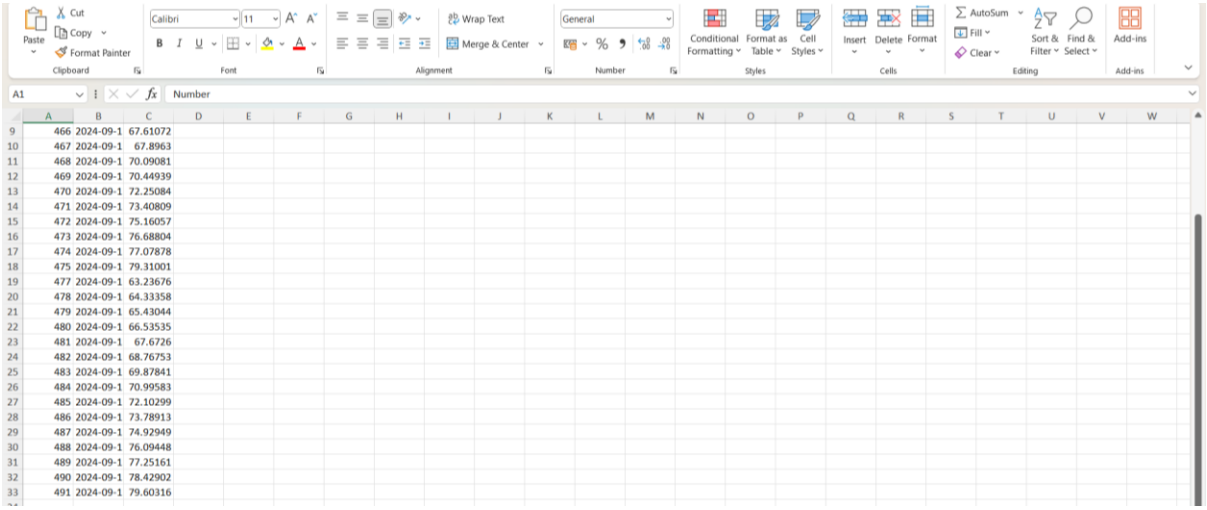

Fig.6 Log files

CONCLUSION

This project successfully implements a distributed number printing application with memory monitoring and automatic server switching on memory leaks. By utilizing Redis for data synchronization and Apache POI for logging, it offers a robust solution for both developer and user-centric logging. The API is designed to ensure seamless server switching and user-friendly logging. Future improvements could include more complex memory management and scaling the application across multiple nodes.