

A Report
On
SHOP MANAGEMENT SYSTEM
Submitted to
University of Petroleum and Energy Studies
In Partial Fulfilment for the award of the degree of
BACHELORS of TECHNOLOGY
In
COMPUTER SCIENCE AND ENGINEERING (with
specialization in Cloud computing)
By
Richa Yadav
Sap id:**500097178**
Under the guidance of
Saurabh Shanu sir

University of Petroleum and Energy Studies
Dehradun-India
November 2023

CLOUD PERFORMANCE TUNING

CONTENT OF REPORT

- 1. Cloud performance tuning basics.....**
- 2. Problem statement.....**
- 3. Background.....**
- 4. Objective.....**
- 5. Sub objective.....**
- 6. Methodology.....**
- 7. Theoretical framework.....**
- 8. Source of data.....**
- 9. Schematic diagram.....**
- 10. Literature review.....**

Cloud Performance Tuning Basics

With increase in adoption of cloud computing and more and more application being deployed in cloud , performance tuning becomes more and more integral part of how business operate today be it in terms of flexibility, scalability and cost effectiveness . To make the best out of cloud computing , performance tuning and optimization of code is very important.

What do we mean by performance tuning ?

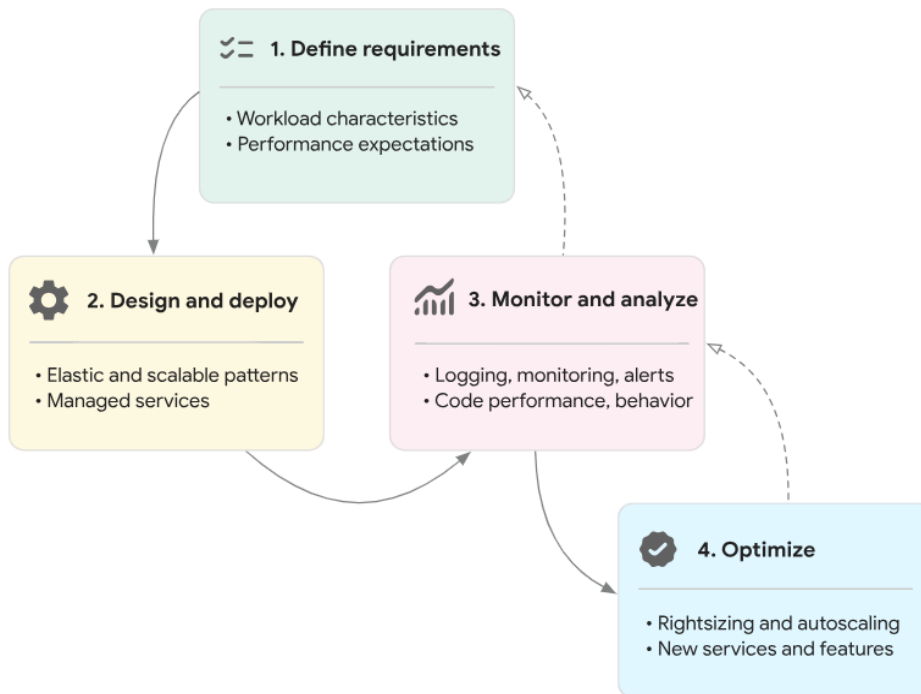
It is the improvement of system's performance. Mostly performance is tuned for an application to improve **scalability** , scalability is the ability of system to handle larger load with efficiency.

Now performance can tuned in many ways. However when it comes to cloud based applications it becomes an easier task to perform ,Azure itself provides various monitoring and diagnostic tools to observe and instrument performance.

It is done to ensure high availability, meet SLAs, and provide seamless cloud services to users.

The introduction basically includes the importance of cloud monitoring and performance optimization in the context of ensuring high availability of cloud-based systems. Cloud monitoring involves continuously assessing the health, performance, and availability of resources while aiming to maximize resource efficiency.

Following are the basic steps to follow for tuning application



1. We need to define the requirements : frontend load balancing, web or applications servers, database, and storage.
2. Deploy application
3. We have to continuously monitor performance by using logs and alerts, analyze the data, and identify performance issues and take snapshots
4. Based on the performance of applications and changes in requirements, configure the cloud resources to meet the current performance

Common performance optimization strategies in cloud environment include

- resource allocation
- load balancing
- auto-scaling,

One can tune various aspects of a system that is :

- Compute
- Storage
- Network
- Database
- cost

Problem Statement

Creating a shop manager system to help with efficient functioning of the shop its deployed in. Its main motive is to decrease manual recording proces and digitalize it hence saving resources and time of customer and shopkeeper.

It granularly takes care of stock management and keep customers tracking history in record.

In reference to performance : there are mainly 2 problems which arose during the run

Scenario in hand :

1. When customers increased , backend could not keep up with the tracking history. Although, more and more customers were successfully being added. Hence there is some discrepancy
2. System would not give alert in case of inventory being down (multi services failure)

Background

Creating a shop management system to help with the efficient functioning of the shop it's deployed in. It provides necessary features to help various entities. There are 3 profiles in the application: customer, shop inventory, and shopkeeper.

Customers and shopkeepers are for smoothing transactions.

Shop inventory profile would help keep track of stocks and customer demands. The inventory tracking can be used to maximize profits by identifying the most and least sold products.

The customer profile is for each individual customer to add things to the cart. Storage of cart details. Gives unique identification for each customer, and quick payment options.

Shopkeeper profile contains shop id to identify stores uniquely, transaction IDs. Details about the items sold on a particular day on his shift and details about the items sold at his store. Keeping track of transaction logs by date.

KEY FEATURES :

inventory management, sales, customer management, and reporting

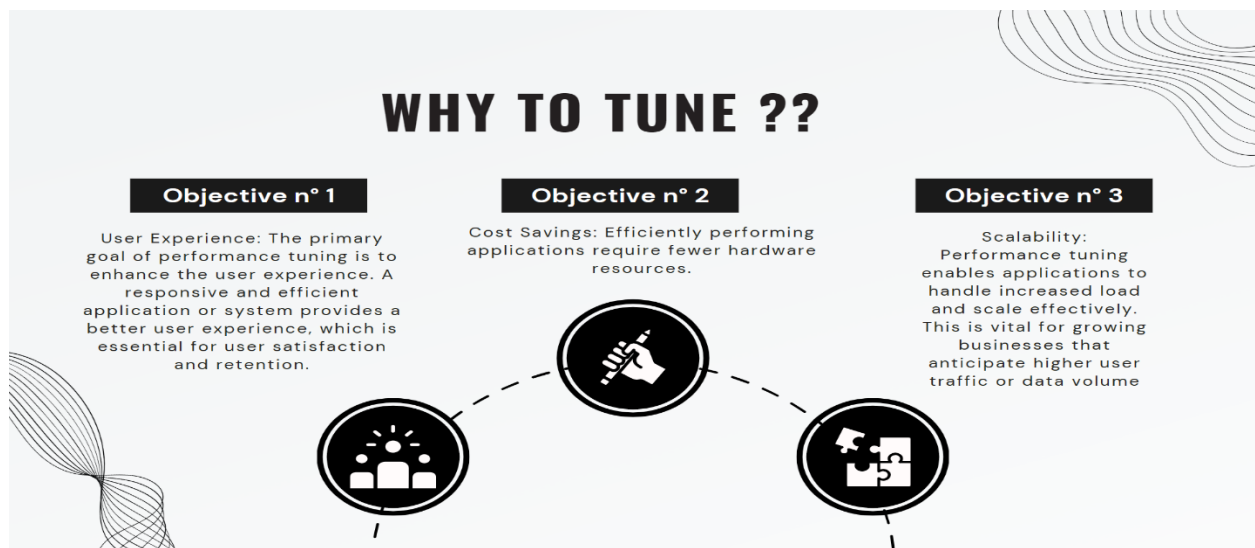
- Provide shop owners with a reliable, user-friendly, and efficient software solution to manage their operations.
- Improve inventory management and stock control, reducing losses due to overstocking or understocking.
- Enhance the customer experience by keeping track of their preferences and purchase history.
- Enable data-driven decision-making through robust reporting and analytics.

Motivation/need for the CPT

Performance profiling is a vital role player in optimizing the system's performance. It mainly focuses on meeting below characteristics:

- SLA (service level agreement) : which are basically deals or guarantee paper signed between cloud vendor and cloud consumer to ensure there is high availability (99.999), zero down time etc.
- QoS (quality of service) : smooth business transactions
- User experience : by tuning system continuously, it impacts on the customer side and provides a healthy interface

- Scalability : with increase in load and demand , system must be able to work effeciently as it was behaving earlier , hence meet reliable and scalaable demands
- Precise use of resources : right resource allocation
- Cost saving : Because autoscaling keeps the resources from being drained, it helps to deliver predictable performance when the demand grows. By eliminating idle resources, autoscaling also aids in cost reduction during times of low load.



Objective

1. To create a java based application on inventory management of shop which is reliable for shopkeepers.
2. Connect it to database , managing queries.
3. Run it without tuning in visual code community , as it is
4. Take record of its performance
5. Tune the code , hardware wise , software wise and network wise
6. Simulate azure environment and try changing configurations and Optimize code by reducing LOC
7. Run it again and observe the changes , if they are useful keep otherwise discard them off.

GOALS AND OBJECTIVES

Objective n° 1

to create a working application in java

Objective n° 2

check performance tuning with help of vs code community and alter code and check again for most optimal solution

Objective n° 3

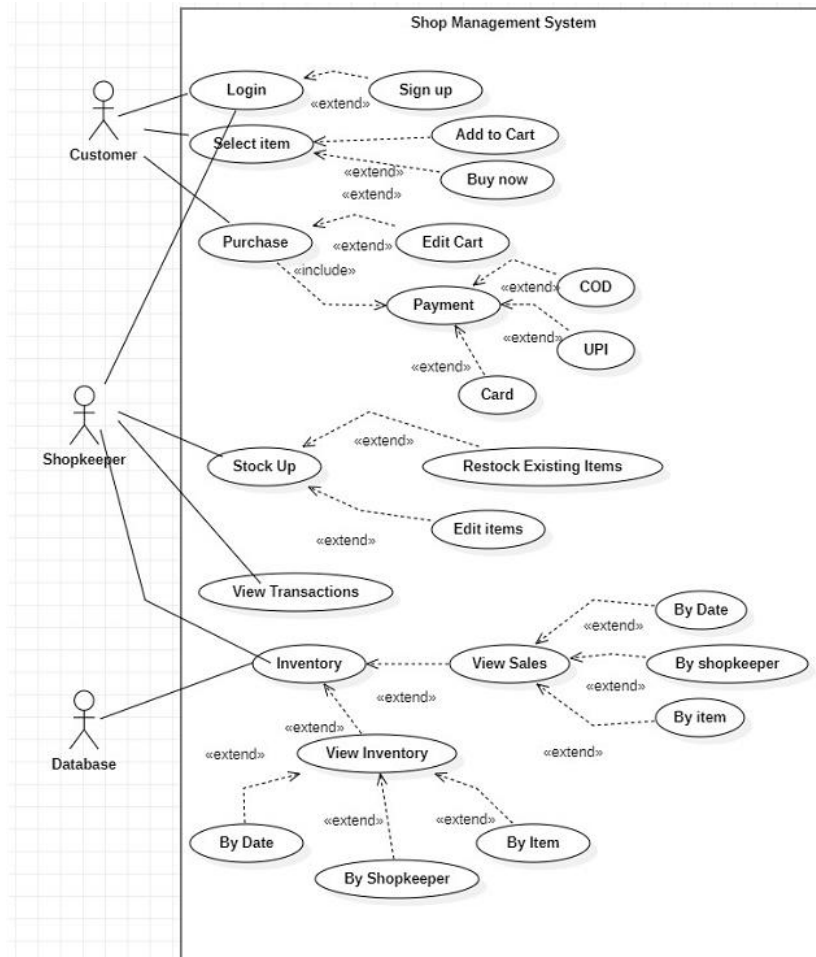
integrate project with cloud on Azure platform and then monitor performance tuning



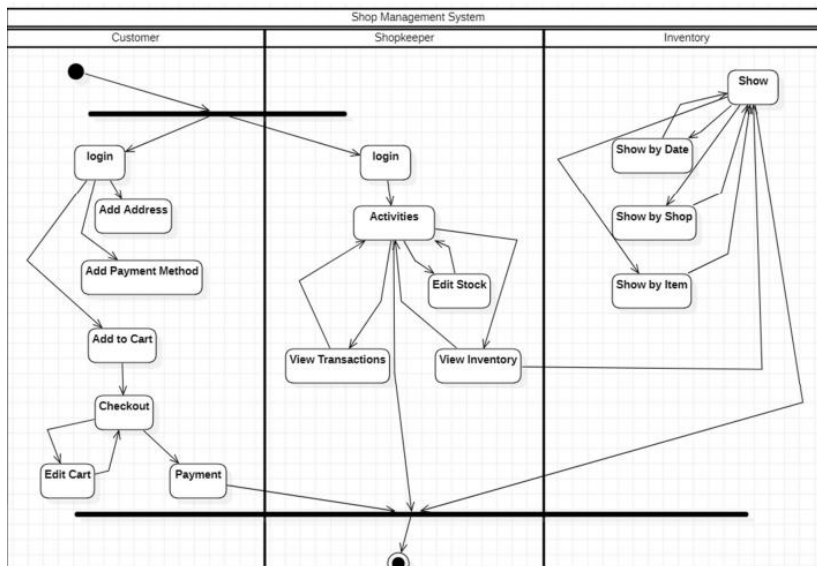
Sub-Objectives

1. Creating frontend and backend both for the application in java by using awt library / package

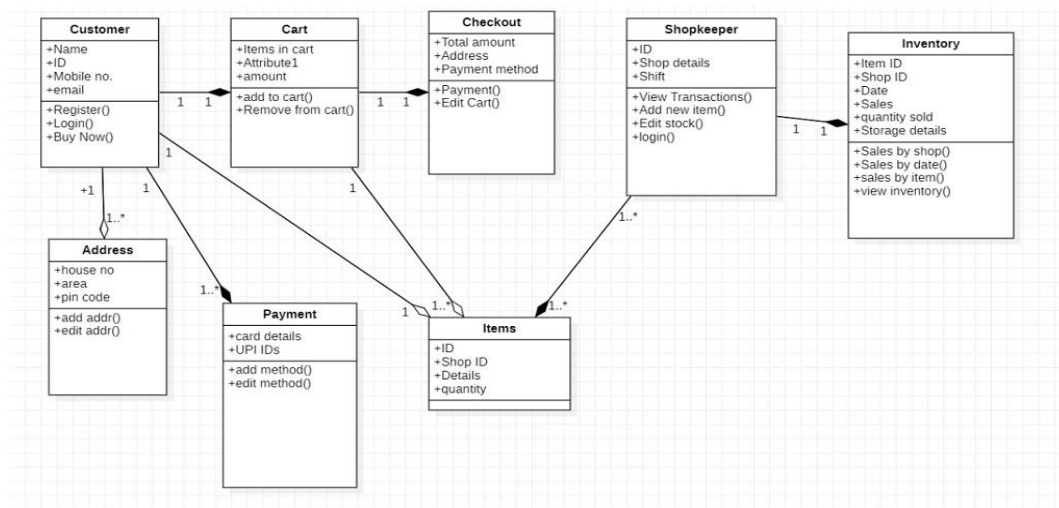
Design -1 (Use Case and Class Diagram)



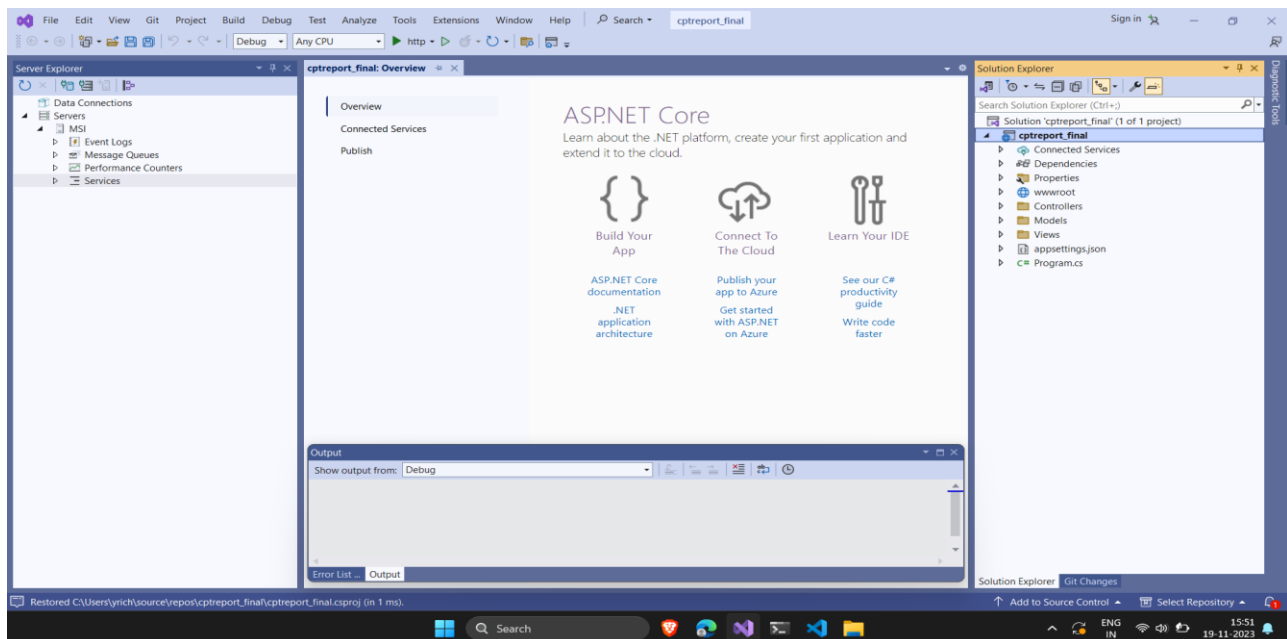
Design -2 (Activity Diagram and State Diagram)



Class Diagrams:



2. . generating reports and dashboards to track customer movements, identify areas for improvement, and make data-driven decisions.
3. Designing a user-friendly interface that simplifies navigation, task completion and easy for noob customers.
4. Integrating the system on azure simulation , a feature in visual studio and publish this.



5. Regularly updating the system by recognizing bottlenecks and features to maintain optimal performance and prevent potential vulnerabilities.

This can be done using diagnostic tools and performance profiler and others such as data optimization tools, load testing tools etc.

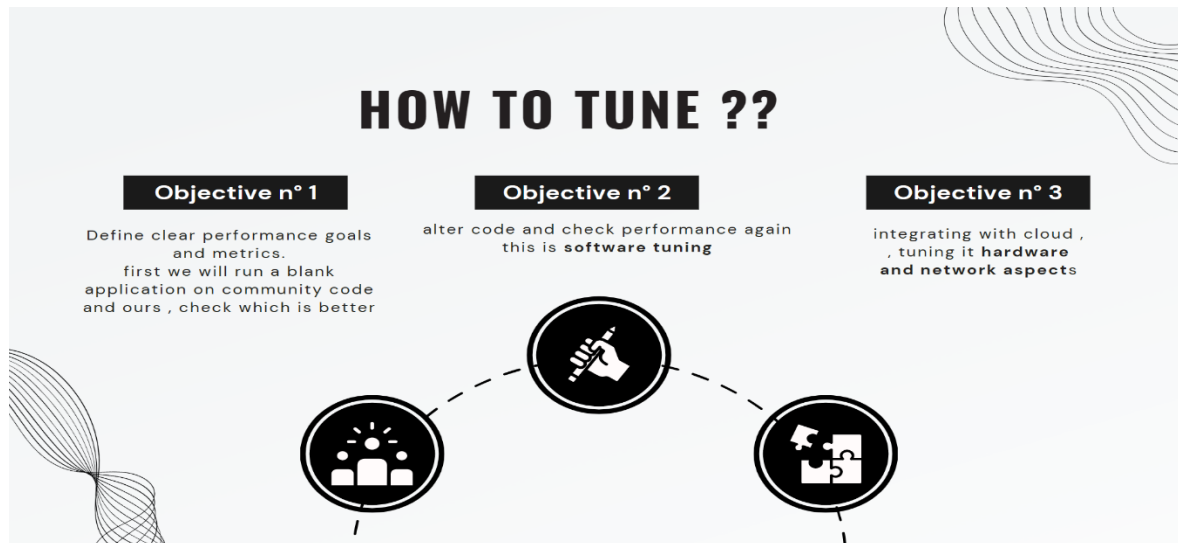
Methodology

Following things have been done

- used arrays and string to store customer name and their profile , plus exception handling which is plus point of java has been executed in every code.
- SQL queries have been written to create tables and insert customer and product info in it , so that one can fetch it whenever customer request a call.
- the View component contains all the UI logic of the application. The components include buttons ,task bar etc
- Single Class Responsibility In SOLID, the first letter talks about Single Class Responsibility. This principle states that “A class should have and only one responsibility to fulfil and only one reason to change” this principle makes a lot easier to implement and prevents unexpected side effects in case of any changes made to the class in the future

Now coming to viewpoint of performance :

“Traditional Java web-based applications are designed for high concurrency and low latency. However, these optimizations may not work well in Cloud based environment, which only serve requests when actively serving.”[1]



I have adopted prioritized Tuning optimization which is based on the impact of each bottleneck on overall system performance

Why ?

As my application is a small one and there are hardly 2 major bottlenecks in it , so it became easy to identify and fix which might not be the case with dynamic kind of apps.

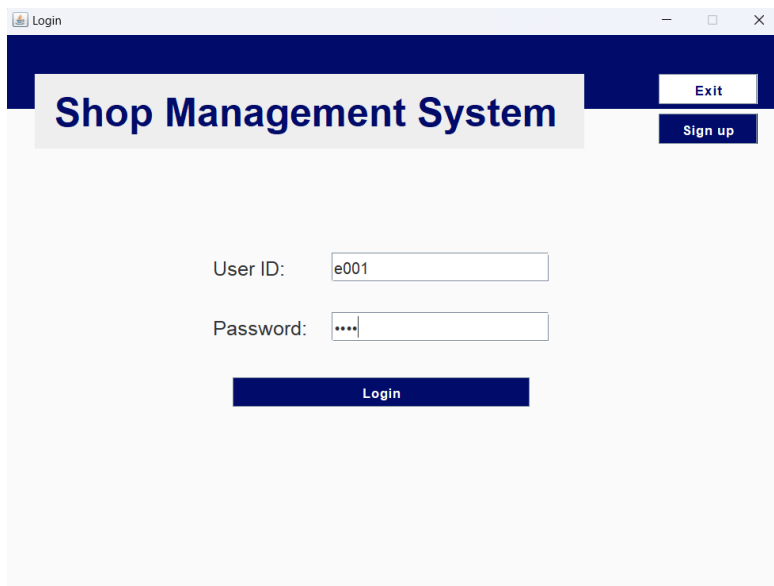
. I chose this one over the other two:

- iterative Approach which involves identifying bottlenecks, implementing optimizations, re-measuring performance, and repeating the cycle until desired performance levels are achieved.
- Data-Driven Optimization which depends on data and performance chart rather than guessing game

There are generic steps which are used to do program profiling:

1. Assessing the problem and setting performance goals like reducing lagging time and faster generation of responses.
2. Measuring the performance of the system before modification Like cpu usage , memory process etc
3. Identifying the part of the system that is critical for improving the performance. This is called the **bottleneck**.
4. Modify that part of the system to remove the bottleneck.
5. Measure the performance of the system after modification. If the modification makes the performance better, adopt it otherwise discard it.

APPLICATION SCREENSHOTS :



The screenshot shows a web browser window titled "Login". The page has a dark blue header with the text "Shop Management System" in white. To the right of the header, there are two buttons: "Exit" (white with blue text) and "Sign up" (dark blue with white text). Below the header, the main content area is light gray. It contains two input fields: "User ID:" with the value "e001" and "Password:" with masked characters "....". Below these fields is a dark blue button labeled "Login".

View Product

View Product

Logout

Back

Add

Keyword: By ID

PID	Name	Price	AvailableQuantity
00001	Juice 1L	66.0	7
00002	Mi Band	2000.0	5
00003	Frutika	55.0	21
00004	Samsung S9	89999.0	5
00005	Bagpack XL	2500.0	6

View Customer

View Customer

Logout

Back

Keyword: By ID

CustomerID	CustomerName	PhoneNumber	Address
c001	Customer	+8801234567890	banani
deba	Debashish	+8801763923789	Kuril

Purchase History

Purchase History

Logout

Back

PurchaseID	ProductID	ProductName	Amount	Cost	Date
00002	00005	Bagpack XL	1	2500.0	2018-09-28
00006	00003	Frutika	2	110.0	2018-09-28

My Profile

My Profile

Logout

Back

Change Password

Delete Account

User ID: c001

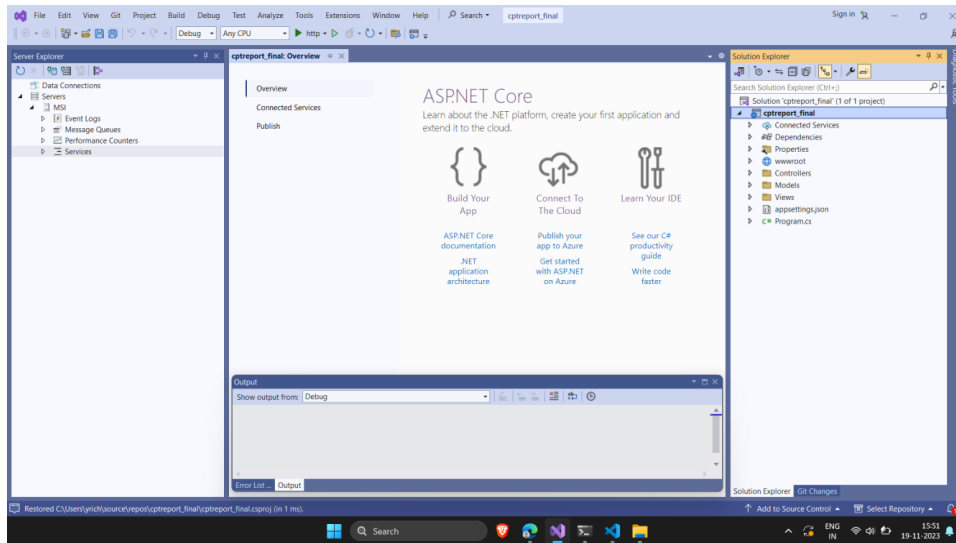
Name: Customer

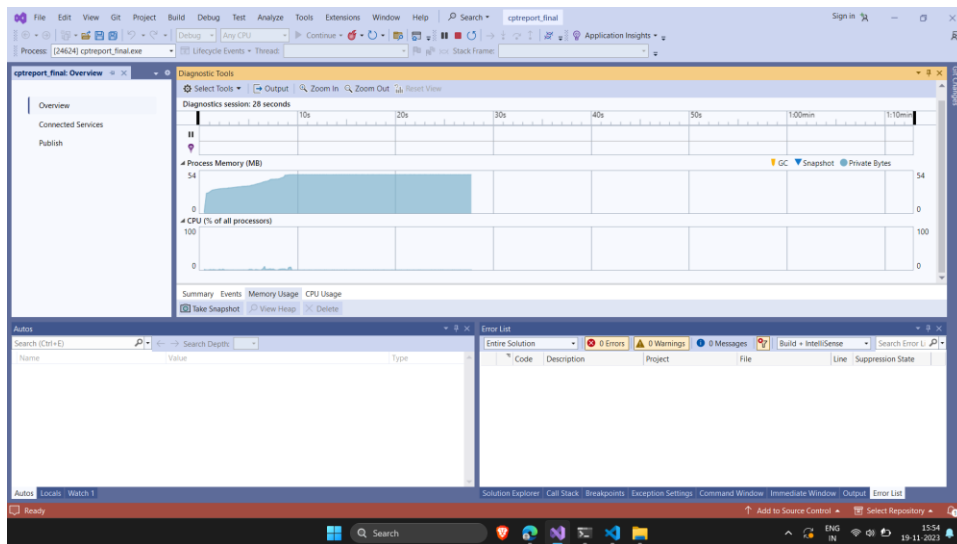
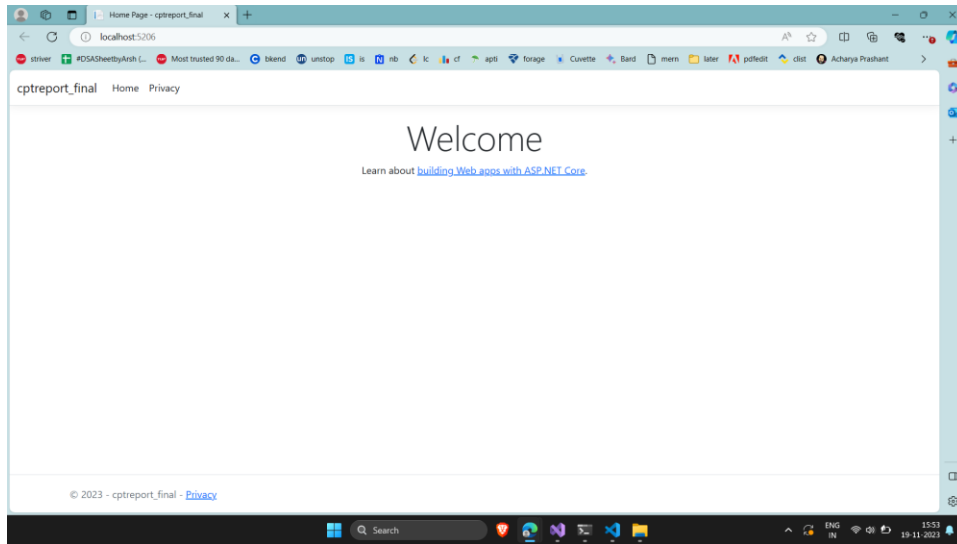
Phone No: +880 1234567890

Address: banani

Edit Profile

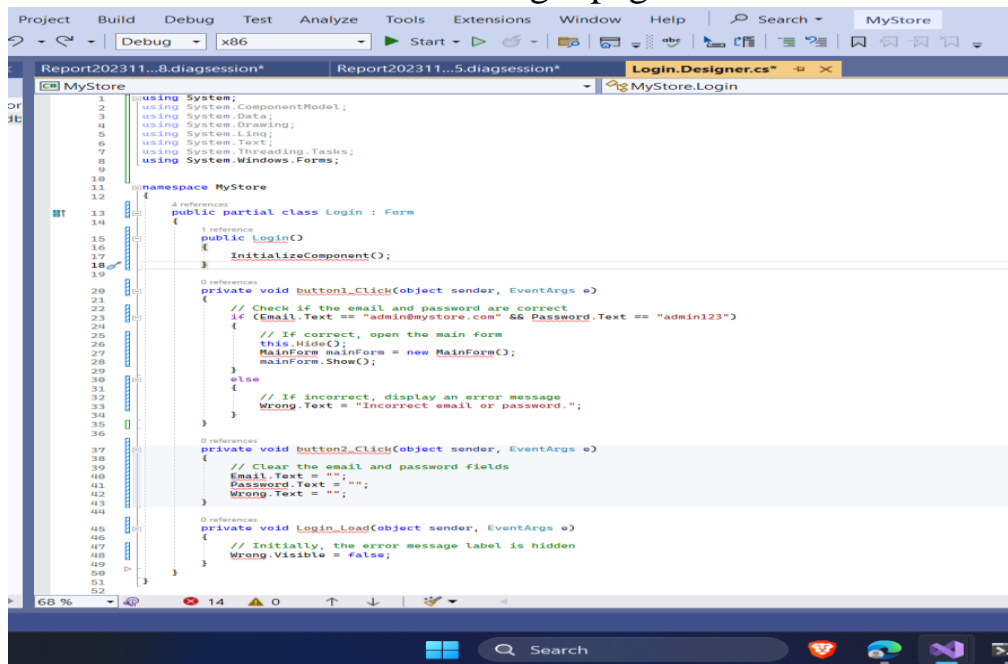
TUNING SCREENSHOTS (test)





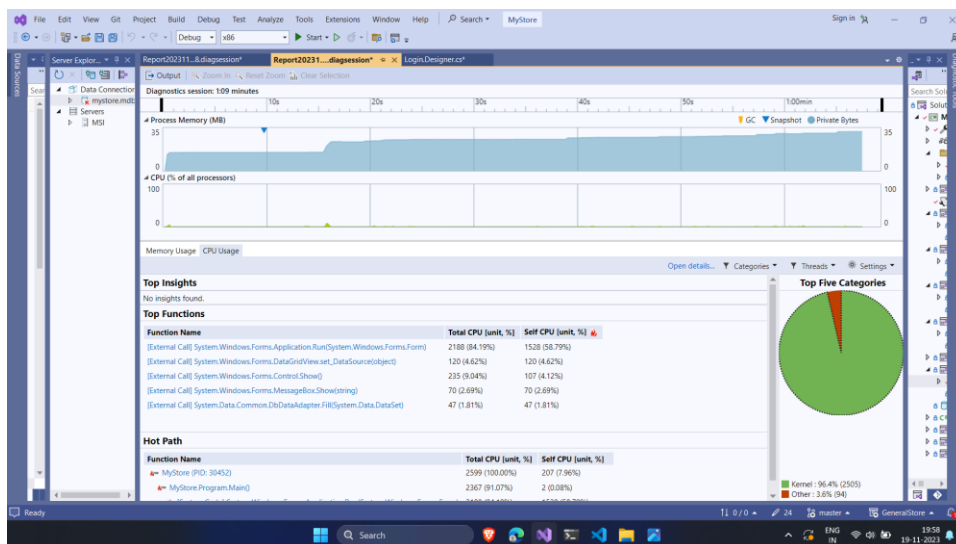
Under this scenario I have one example that can easily illustrate how I have tuned the performance via code optimization

1. Below is the untuned code for login page in .cs



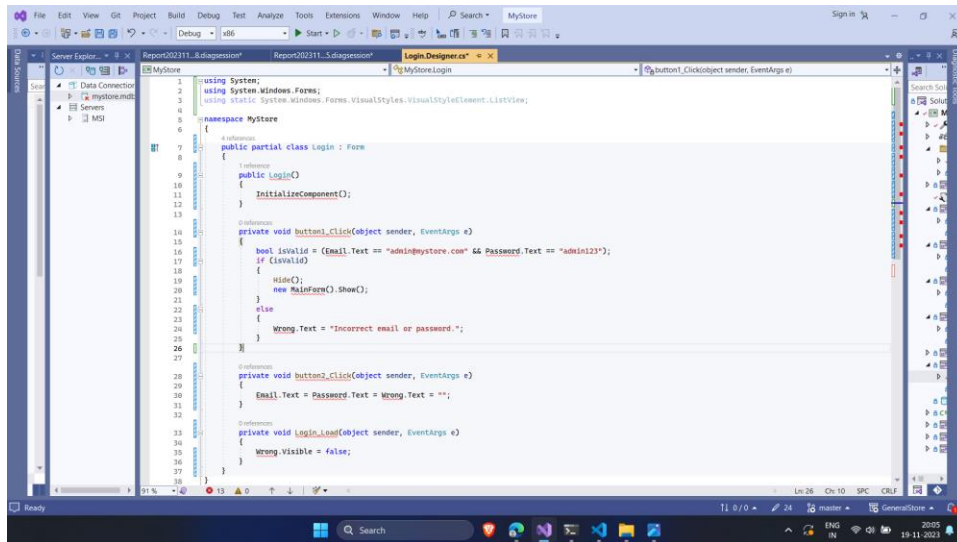
```
1 using System;
2 using System.ComponentModel;
3 using System.Data;
4 using System.Drawing;
5 using System.Linq;
6 using System.Text;
7 using System.Threading.Tasks;
8 using System.Windows.Forms;
9
10 namespace MyStore
11 {
12     public partial class Login : Form
13     {
14         public Login()
15         {
16             InitializeComponent();
17         }
18
19         private void button1_Click(object sender, EventArgs e)
20         {
21             // Check if the email and password are correct
22             if (Email.Text == "admin@mystore.com" && Password.Text == "admin123")
23             {
24                 // If correct, open the main form
25                 this.Hide();
26                 MainForm mainForm = new MainForm();
27                 mainForm.Show();
28             }
29             else
30             {
31                 // If incorrect, display an error message
32                 Wrong.Text = "Incorrect email or password.";
33             }
34         }
35
36         private void button2_Click(object sender, EventArgs e)
37         {
38             // Clear the email and password fields
39             Email.Text = "";
40             Password.Text = "";
41             Wrong.Text = "";
42         }
43
44         private void Login_Load(object sender, EventArgs e)
45         {
46             // Initially, the error message label is hidden
47             Wrong.Visible = false;
48         }
49     }
50 }
```

2. Performance of CPU usage and process memory under this code

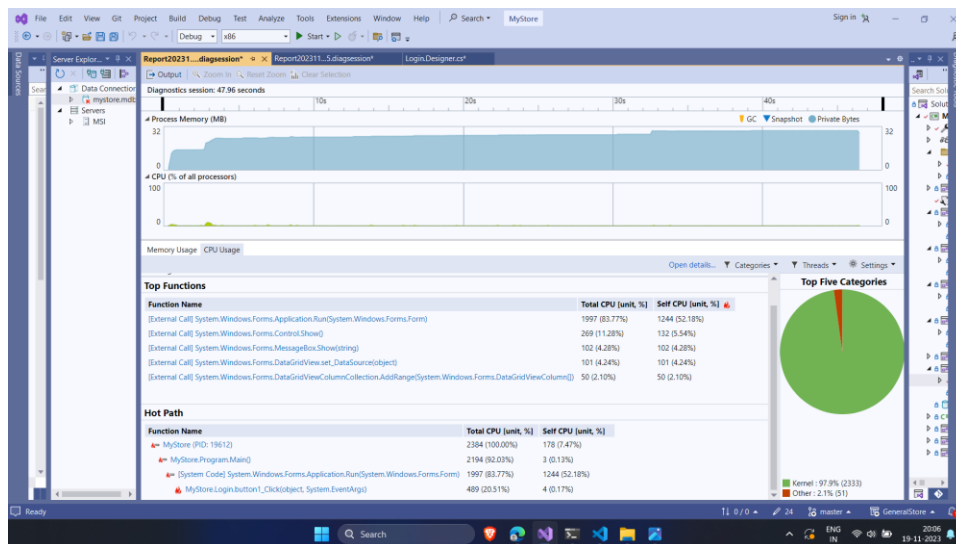


3. And now optimized code: which has reduced LOC because

- complex if-else block has been replaced with a simpler isValid variable.
- Used string interpolation in the new MainForm() call to make the code more concise.
- Have Combined Email, Password, and Wrong fields into a single statement



4. Here are the results , it is evident from pie chart and graph under memory process that cpu usage has been decreased to 83.77% from 84.19%.



Diagnostic session : 47.96 sec

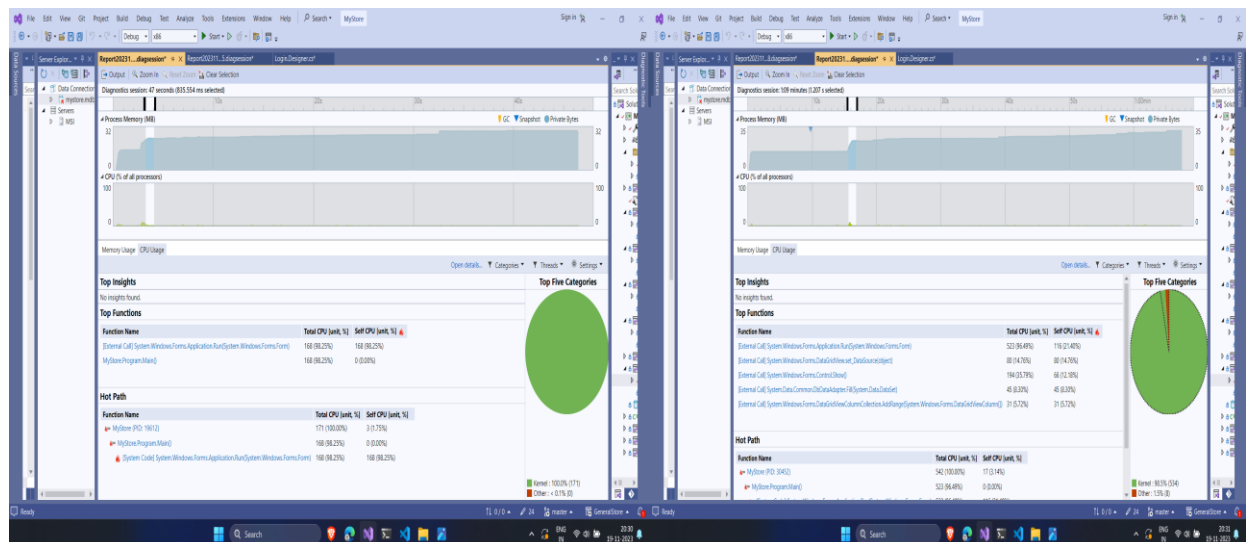
Learnings: (same is the case for memory usage)

SCENARIO	1 st case (unoptimized)	2 nd case (optimized)
1. LOC	51	38
2. When new users are added simultaneously (memory process)	For just one user, app is easily handling the load but when users are added faster and parellely transaction is being done , graph peaks and comes to a threshold	For same situation , the graph peaked at initial stage and held on to it , until transaction being made and performed to its peak effeciency
3. CPU usage	83.77%	84.19%

Sudden load increase case : @2.87 seconds

Unoptimized : When there is spike in customer activity , 100 % of kernel was being used

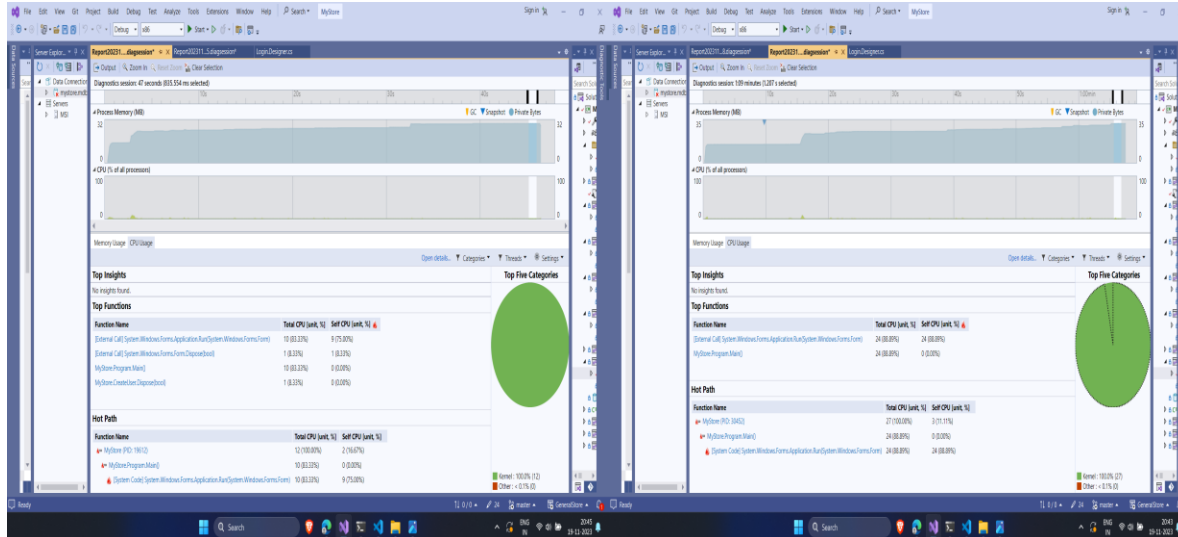
Optimized : When there is spike in customer activity , 98 % of kernel was being used



Peak load : @46.97 sec

Unoptimized : total CPU units used : 88% (24 units)

Optimized : total CPU units used : 83% (10)



Theoretical framework

Why java ?

- Platform Independence: "write once, run anywhere" .
- Scalability
- Security

Java swing: It is used for creating window-based applications. It includes components like buttons etc and makes a graphical user interface.

Java AWT: Java AWT (Abstract Window Toolkit) is an API to develop Graphical User Interface (GUI) or windows-based applications in Java.

Mysql: MySQL is a relational database management system based on SQL – Structured Query Language.

- Performance optimization techniques involve optimizing database queries and code to reduce complexity, optimize algorithms, and minimize memory allocations, thereby enhancing efficiency and reducing startup time..
- One can always automate the testing and deploying process, as running the application during test case is very different from running it in real time environment. So if tuning is done parallelly during execution of code, then it would increase efficiency.
- Hence, I could implement CI/CD pipeline through Jenkins and automate the process of developing and deployment, so that maintenance is done.
- While tuning performance, I need to make sure of how the user experience is getting affected, and whether cost is being compromised? They should all find balance in each other.

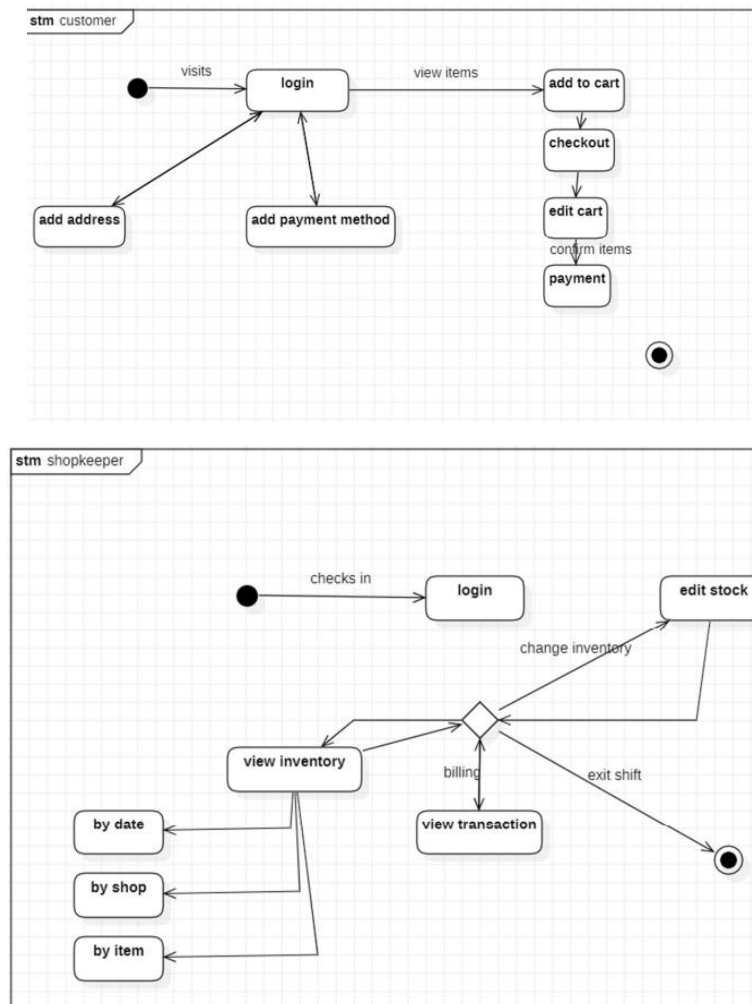
Sources of data

Primary or secondary data:

1. [Performance analysis - Wikipedia](#)
2. [59860.pdf \(scitepress.org\)](#) for java app
3. [Optimize Java applications for Cloud Run | Cloud Run Documentation | Google Cloud](#)
4. [Performance Diagnostics Tools | Microsoft Learn](#)

Schematic flow Diagram

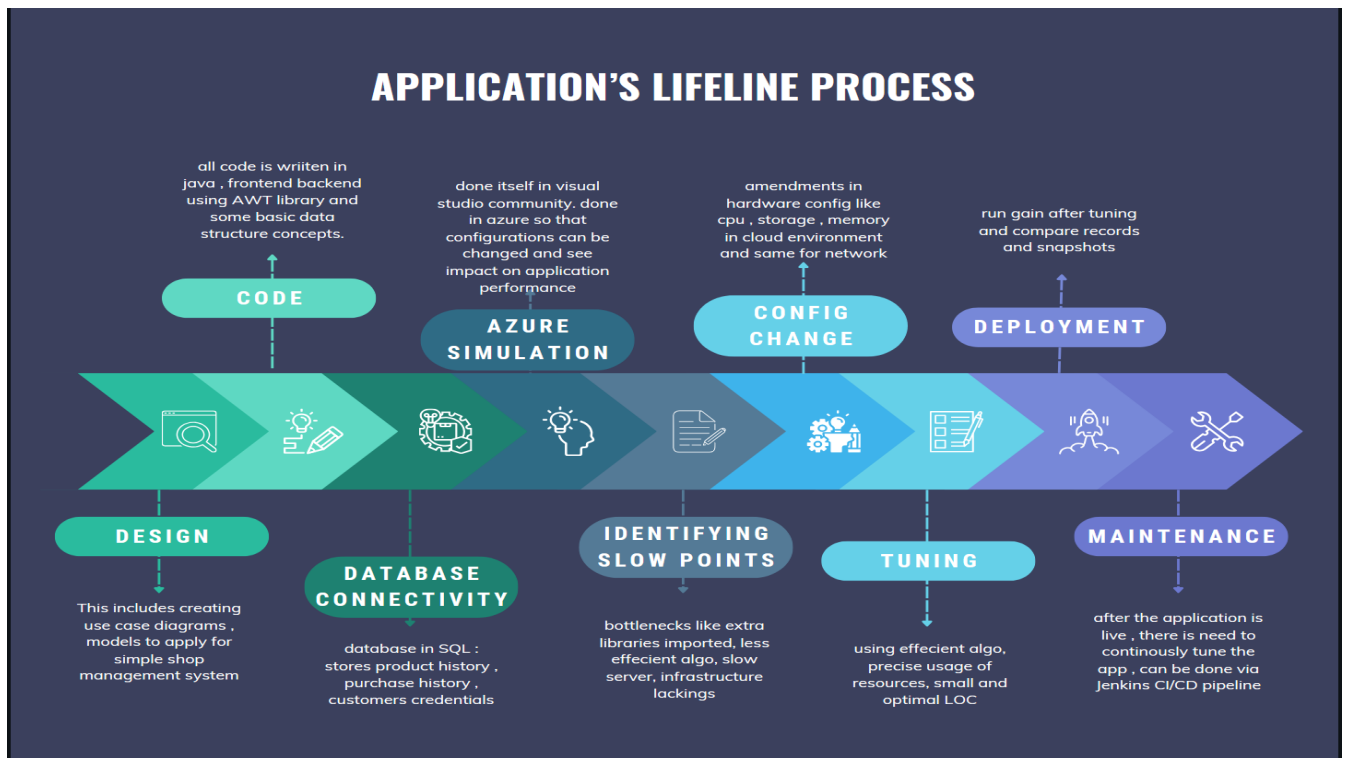
For application design and code



It is clearly mentioned in below flow chart , how I have done tuning in my application

This diagram clearly states the process from scratch:

1. Performance tuning begins right from designing and architectural phase that is designing an optimal , low LOC code right from the beginning.
2. Coding phase : keeping the lines of code short and crisp , like replacing higher complexity algo with lower complexity one in terms of time and space.
3. Database queries also are written in short manner avoiding extra joins and doing proper indexing.
4. Deploying it in .NET azure framework
5. by analyzing code execution paths, database query patterns, and resource utilization patterns we can see application's performance , which could also be visible from UI while using it
6. Tuning It by doing proper indexing in database and optimizing code and updating hardware resource to divide load and improve response times.
7. Deploying it and checking the graph now and making comparison
8. Continuing the health checkup by automation (this one is additional ; not implemented yet)



Review of literature

The literature on cloud monitoring and performance optimization underscores the significance of maintaining high availability and optimal performance

1. Importance of Cloud Monitoring: . Effective monitoring practices allow businesses to meet service level agreements

(SLAs) and ensure consistent user experience

2. Strategies for Performance Optimization: Efficient resource allocation, load

balancing, and auto-scaling are highlighted as key techniques to distribute workloads evenly

across cloud resources, preventing resource contention and ensuring optimal performance.

3. Impact of Performance Optimization on Cloud Services: auto-scaling dynamically adjust resource allocation based on workload demands, ensuring that applications can scale up or down as needed, resulting in cost savings and enhanced performance.

4. Challenges in Cloud Monitoring and Performance Optimization: , ensuring data security and privacy is critical, as

monitoring data may contain sensitive information.

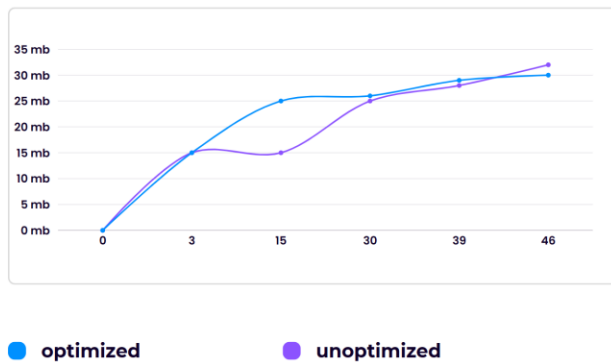
Real-world Applications and Case Studies: it has a visible impact on application as we also observed in this application of shop management system , when customer or stocks fluctuated and how performance tuning solved the issue in hand

Result

The implementation of cloud monitoring and performance optimization has yielded significant results in ensuring that there is no lag in taking input and processing.

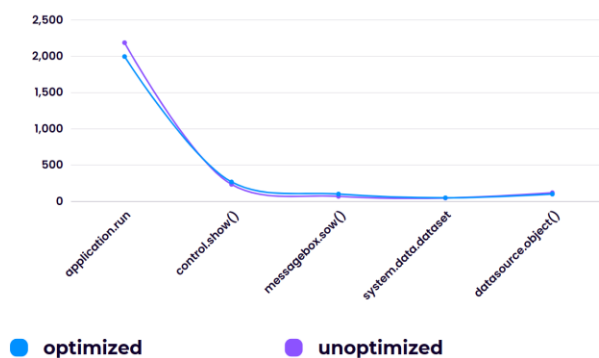
Through this I could detect and address problems and maintain consistency of app.it was a challenging task to have done under c sharp as I had code in java , that is why code optimization was bit difficult . However, I could successfully do the load testing and eliminate the lag between input and processing via decreasing line of code and using less memory for processing.

**process memory usage
comparision**



From the graph it is evident that , process memory was used more by unoptimized section (blue line) at sudden spike in load or consumer activity , however optimized section (purple line)also used it but always had an average edge over the prior one, even at peak period

CPU usage comparison



Very clearly stated in the above graph , CPU utilization is slightly higher in unoptimized code. This graph has been generated by following data , got during performance profiler

function name	unoptimized	optimized
application.run	2188	1997
control.show()	235	269
messagebox.so	70	102
system.data.da	47	50
datasource.obj	120	101

Here are the final comparable result therefore I saved resources and optimized my application using cloud performance tuning , as all of it is done in Azure simulation.

LINK TO GITHUB: <https://github.com/pixlricha/Cloud-performance-tuning.git>

THANK YOU !

List of Figures

Cloud Performance Tuning Basics

Problem Statement

Background

Motivation/need for the CPT:

Objective

Sub-Objectives

Mode of achieving objective

Methodology

Theoretical framework – explains the model or the set of theories related to the CPT.

Sources of data – Primary or secondary data:

Schematic flow Diagram:

Review of literature

Key Bibliography