NM1051 – SERVICENOW ADMINISTRATOR

STREAMING TICKET ASSIGNMENT FOR EFFICIENT SUPPORT OPERATION

A PROJECT REPORT

Submitted by

AROCKIYA JERMILA A	**-**	**962722104008**
CHITHRA P	**-**	**962722104017**
NANCY C	**-**	**962722104033**
SRI THARSHINI K	**-**	**962722104048**

BACHELOR OF ENGINEERING

IN SEVENTH SEMESTER

COMPUTER SCIENCE ENGINEERING

UNIVERSAL COLLEGE OF ENGINEERING AND TECHNOLOGY

VALLIOOR – 627117

ANNA UNIVERSITY: CHENNAI 600025 /DECEMBER -2025

BONAFIDE CERTIFICATE

Certified that this project "**STREAMING TICKET ASSIGNMENT FOR EFFICIENT SUPPORT OPERATION**" is the bonafide work of **AROCKIYA JERMILA A (962722104008), CHITHRA P** (**962722104017**), **PAVITHRA N (962722104037), SRI THARSHINI K (962722104048),** who carried out the project work under any supervision.

SIGNATURE.

Prof.M.Pradeesh Kumar., ME.,

HEAD OF THE DEPARTMENT,

Dept. of Computer science Engg

Universal College of Engg &Tech
Vallioor - 627117

SIGNATURE.

Prof.M.Chandralekha., ME.,

SUPERVISOR,

Dept. of Computer Science Engg

Universal College of Engg & Tech
Vallioor - 627117

Submitted For the Anna University Examination held on……

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

We sincerely thank to “Naan Mudhalvan” Platform and for encouragement towards our project work for providing necessary skill training.

We sincerely thank our principal **Dr.T. ASEER BRABIN, ME., MISTE., PHD.,** for encouragement towards our project works.

We also thank our He of the Department and our project guide and our parents for the complete and wholehearted support, motivation guidance and help in making our project activities

Table of Contents

Abstract

In modern customer service and technical support environments, efficiency and speed are crucial in maintaining customer satisfaction and operational effectiveness. As organizations grow and user interactions increase, managing and resolving support tickets promptly becomes increasingly challenging. Traditional ticket assignment systems often rely on manual intervention or static rule-based methods, which can lead to delays, misallocation of resources, and inconsistent service levels. To address these challenges, this project proposes a **Streaming Ticket Assignment System** designed to enhance the efficiency and accuracy of support operations through dynamic, real-time ticket allocation using intelligent algorithms and automated workflows.

The primary objective of this project is to streamline the ticket distribution process in helpdesk or support centers by continuously monitoring incoming requests and assigning them to the most appropriate support agent based on parameters such as skill set, current workload, priority level, and ticket complexity. This streaming approach ensures that tickets are not queued unnecessarily and that high-priority issues are attended to immediately. The system reduces response time, balances workloads among support agents, and increases overall customer satisfaction.

In traditional models, tickets are typically assigned in batches or at predefined intervals, causing inefficiencies, especially during peak hours. These models fail to adapt to fluctuating workloads and do not consider real-time operational data. The **Streaming Ticket Assignment System**, however, leverages streaming data processing techniques and machine learning components to assign tickets dynamically as they arrive. It continuously evaluates system conditions and agent availability to ensure that each ticket reaches the right person at the right time. This continuous, event-driven architecture enables real-time decision-making, which is essential for high-volume support operations such as IT service management, e-commerce platforms, and telecom customer care systems.

The proposed system architecture includes several key components: a ticket input module that captures support requests from various channels (emails, chatbots, web forms, etc.), a data streaming layer that handles continuous input data, an intelligent assignment engine that applies algorithmic logic to match tickets with agents, and a monitoring dashboard for

administrators to visualize performance metrics and operational trends. The assignment engine can incorporate algorithms such as weighted round-robin scheduling, load balancing, and skill-based routing. Additionally, it can integrate with predictive models that estimate ticket resolution time and agent efficiency based on historical data.

The project also explores the integration of **Python-based backend development** using libraries such as Flask for web framework implementation and Pandas for data handling. Streaming data handling may be supported through tools like **Apache Kafka or RabbitMQ**, ensuring smooth and real-time message processing. The system's design emphasizes scalability and modularity, allowing it to adapt to varying organizational sizes and dynamic customer service environments. Frontend visualization components can be developed using technologies like HTML, CSS, and JavaScript or integrated with modern frameworks for real-time monitoring dashboards.

One of the significant advantages of the proposed system is its ability to minimize human error and manual oversight. By automating ticket routing and prioritization, it ensures that no request is left unattended. The continuous streaming model also enables predictive analytics, helping support managers forecast workloads and optimize staffing accordingly. Furthermore, by tracking performance metrics such as response time, ticket resolution rates, and agent utilization, the system provides actionable insights that contribute to continuous process improvement.

The project's implementation will be demonstrated through simulation or prototype deployment, showcasing how real-time ticket flow can be efficiently handled and visualized. Performance metrics will be compared against traditional ticket assignment systems to highlight improvements in speed, accuracy, and overall service quality. The outcomes are expected to show a substantial reduction in average response times and better workload distribution across agents, confirming the system's operational benefits.

In conclusion, the **Streaming Ticket Assignment for Efficient Support Operation** project represents an innovative and practical solution for organizations striving to optimize their customer support mechanisms. By leveraging real-time data streaming, automation, and intelligent routing algorithms, the system significantly improves efficiency, reduces delays, and enhances customer experience. This approach aligns with modern digital transformation

goals, where automation, scalability, and responsiveness are central to organizational success. Future enhancements can include machine learning-based predictive allocation, integration with AI chatbots for auto-resolution of simple issues, and analytics-driven decision support for management teams. Overall, this project contributes to the growing domain of intelligent operations management by presenting a scalable, data-driven, and efficient ticket assignment solution for the modern support ecosystem.

Introduction

In the modern digital era, customer support and service operations have become critical pillars of organizational success. With the growing dependence on digital platforms, online services, and remote operations, organizations receive thousands of customer requests, complaints, and inquiries every day. Efficiently managing and resolving these requests—commonly referred to as "tickets"—has a direct impact on customer satisfaction, business continuity, and brand reputation. However, as customer bases expand and service demands increase, manual or static ticket assignment systems often fail to keep up with the pace, leading to delayed responses, unbalanced workloads among agents, and reduced service quality.

To address these challenges, the concept of **Streaming Ticket Assignment for Efficient Support Operation** has been introduced. This project focuses on automating and optimizing the process of assigning support tickets dynamically using real-time data streaming and intelligent decision-making algorithms.

In traditional customer support environments, tickets are typically assigned through manual triaging or rule-based batch processes. For example, an administrator may review new tickets periodically and assign them to available agents based on priority or category. While this method works for small teams, it becomes inefficient and error-prone as the number of requests increases. Delays in assignment, human errors, and lack of visibility into real-time agent availability can lead to longer response times and reduced productivity.

Modern organizations need systems that can process large volumes of support requests in real time and allocate them efficiently among support personnel. The **streaming ticket assignment system** provides a solution by continuously monitoring incoming tickets and automatically routing them to the best-suited agent as soon as they arrive. This reduces waiting time, ensures fair workload distribution, and enhances operational efficiency.

The motivation behind this project lies in the growing demand for **automation and intelligence in IT service management (ITSM)** and **customer relationship management (CRM)** platforms. By integrating real-time data processing, machine learning models, and intelligent

routing algorithms, support teams can achieve faster resolution times and improved user satisfaction while minimizing operational costs.

Methodology

The **methodology** of the project "Streaming Ticket Assignment for Efficient Support Operation" focuses on developing a system that can dynamically assign support tickets to agents using real-time streaming data and intelligent automation. The methodology defines the **system architecture, process flow, algorithms, and tools** used to achieve the project objectives.

It combines **data streaming technology**, **Python-based backend development**, and **intelligent decision-making algorithms** to ensure that every incoming ticket is immediately processed, analyzed, and assigned to the most suitable agent without manual intervention. The proposed approach consists of several key phases: **requirement analysis, system design, data flow modeling, implementation, and testing.**

3.1 System Overview

The proposed system is designed to manage customer support operations efficiently by automating ticket distribution. The streaming ticket assignment model ensures that as soon as a customer submits a support request (ticket), it is automatically routed to the best-fit support agent.

The system continuously monitors several factors such as:

- **Agent workload** (number of active tickets per agent),
- **Agent skillset** (domain expertise or issue category),
- **Ticket priority** (critical, high, medium, or low), and
- **Current system load** (real-time performance metrics).

Using these parameters, the system applies an **assignment algorithm** that ensures each ticket is processed instantly and fairly distributed among agents.

3.2 Workflow of the System

The workflow of the **Streaming Ticket Assignment System** can be divided into the following stages:

Stage 1: Ticket Generation and Input Stream

Customers submit their support requests through multiple channels such as chat, email, web portals, or mobile applications. These incoming requests are captured in real time and transformed into structured ticket objects. Each ticket contains metadata like ticket ID, issue type, priority, timestamp, and customer details.

A **data streaming layer** (using technologies like Apache Kafka or RabbitMQ) continuously receives these tickets and makes them available for processing. This streaming approach ensures that tickets are handled the moment they are created—without waiting for batch processing.

Stage 2: Preprocessing and Classification

Before assigning, each ticket is analyzed to determine its category and priority level. For example, technical issues may go to Level-2 agents, while billing issues may go to financial support teams.

In this stage, the system may use **Natural Language Processing (NLP)** or **keyword extraction** to classify tickets based on content. The ticket attributes are then enriched with classification tags such as “Network Issue,” “Software Bug,” “Payment Problem,” or “General Query.” This ensures accurate routing in later stages.

Stage 3: Agent Availability Monitoring

The system continuously tracks all available support agents and their current workloads. Each agent’s data includes:

- Agent ID and skill categories
- Number of active tickets
- Average resolution time
- Performance rating
- Online/offline status

This data is updated in real time and stored in an **agent availability database**. The streaming system continuously consumes this data to make decisions for incoming tickets.

Stage 4: Assignment Algorithm

The core logic of the project lies in the **ticket assignment algorithm**. This algorithm evaluates both ticket and agent data to determine the best possible match. Some algorithms that can be used include:

- **Round-Robin Assignment:** Sequentially assigns tickets to agents in a circular order to ensure fair distribution.
- **Weighted Load Balancing:** Assigns tickets based on current workload and agent efficiency. Agents with fewer active tickets receive new ones first.
- **Skill-Based Routing:** Matches ticket category with the agent's area of expertise to ensure higher resolution accuracy.
- **Priority-Aware Assignment:** Ensures that critical or high-priority tickets are assigned immediately to experienced agents.

The chosen algorithm runs continuously in the streaming pipeline, so decisions are made instantly as tickets arrive.

Stage 5: Real-Time Assignment and Notification

Once an assignment decision is made, the ticket is automatically routed to the selected agent's dashboard. Simultaneously, notifications are sent to both the agent and the customer confirming that the request is being handled. This stage ensures there are no delays or manual bottlenecks in the assignment process.

Stage 6: Monitoring and Analytics

A **monitoring dashboard** is developed to visualize ticket flow, agent performance, and system status. The dashboard displays metrics such as:

- Total tickets received
- Average response and resolution times
- Agent utilization rates
- Pending vs. resolved ticket counts

- Priority-wise ticket distribution

These analytics help administrators understand workload distribution and system performance. It also enables predictive analysis for staffing and performance improvement.

4.Existing Work

The management of support tickets has always been an integral part of customer service and IT operations. Over the years, several systems and methodologies have been developed to handle the process of **ticket creation, tracking, and assignment**. Traditional helpdesk and support systems such as **ServiceNow, Zendesk, Freshdesk, Jira Service Management**, and **BMC Remedy** have provided structured platforms for managing service requests. However, despite their effectiveness in organizing customer issues, these tools often rely on static, rule-based mechanisms for ticket allocation and lack real-time automation capabilities.

This section reviews the existing methods, technologies, and limitations of current ticket assignment systems, emphasizing why there is a need for a **streaming-based dynamic approach** to achieve efficient support operations.

4.1 Traditional Ticket Management Systems

In most organizations, support tickets are managed through centralized software systems that allow customers to report issues and agents to track and resolve them. These systems typically include modules for **ticket logging, prioritization, assignment, escalation, and resolution tracking**.

Some of the most widely used traditional ticketing systems include:

- **ServiceNow:** A comprehensive IT service management (ITSM) platform that offers workflows for handling incidents, problems, and service requests. ServiceNow supports rule-based automation but processes assignments in batches or through pre-defined scripts.
- **Zendesk:** Popular in customer support environments, Zendesk routes tickets using triggers and conditions. However, routing decisions are based on static rules and lack real-time workload balancing.
- **Freshdesk:** A cloud-based support system offering ticket categorization and manual assignment options. Although it supports automation rules, it cannot dynamically adapt to agent workload changes.

- **Jira Service Management:** Used widely in software and IT companies for tracking issues and service requests. It provides assignment options but typically depends on predefined rules or manual intervention.

While these systems provide structure and visibility in support operations, their static assignment models often fail under high-volume, dynamic environments where requests arrive continuously.

4.2 Static Assignment Mechanisms

In traditional ticketing systems, **ticket allocation is either manual or rule-based**. The two dominant methods are:

Manual Assignment:

- A team leader or administrator manually reviews each incoming ticket and assigns it to an appropriate agent.
- This approach offers flexibility but is slow, inconsistent, and not scalable for large organizations handling hundreds of tickets per hour.
- Human judgment can vary, and errors in assignment often lead to increased resolution time.

Rule-Based Assignment:

- Rules are predefined based on ticket attributes such as category, priority, or customer type.
- Example: “All network issues go to Team A” or “High-priority tickets go to Senior Agents.”
- While this automates part of the process, it cannot adapt to **real-time workloads** or sudden agent unavailability.
- If all agents in a team are busy, new tickets are still routed to them, causing backlogs.

These static systems, though effective for predictable workloads, do not perform well in dynamic environments where tickets arrive continuously from multiple sources.

4.3 Limitations of Existing Systems

Despite their popularity and maturity, traditional ticket assignment systems have several shortcomings when it comes to efficiency, scalability, and adaptability:

1. *Lack of Real-Time Responsiveness:*
   Existing systems process data in batches or intervals. This causes delays in assignment and reduces responsiveness during peak periods.

2. *Inflexible Rules:*
   Static assignment rules do not adjust automatically to changes in ticket volume or agent workload, leading to inefficiency and unfair workload distribution.

3. *Poor Resource Utilization:*
   Without real-time balancing, some agents become overloaded while others remain idle. This affects team morale and overall performance.

4. *Inability to Predict or Adapt:*
   Traditional systems do not use predictive analytics or machine learning to forecast workloads, identify trends, or adapt routing strategies dynamically.

5. *Manual Intervention Requirement:*
   Even in automated tools, administrative oversight is often required to reassign tickets, resolve conflicts, or balance workloads.

6. *Limited Integration with Streaming Data:*
   Most existing platforms are not designed to handle continuous event streams. They process inputs in queues or batches, which leads to latency and missed opportunities for optimization.

5.Proposed Work

The **proposed work** focuses on developing a real-time, intelligent, and automated **Streaming Ticket Assignment System** to overcome the limitations of traditional ticket management models. The goal of the proposed system is to assign incoming support tickets dynamically to the most appropriate agents using a streaming data pipeline integrated with intelligent decision-making algorithms.

By leveraging **real-time data streaming, automation, and AI-driven decision logic**, the system minimizes human intervention, reduces ticket response time, balances workload distribution, and significantly enhances the overall efficiency of support operations. Unlike static or batch-processing systems, the proposed model ensures **instant assignment** of tickets as soon as they arrive, enabling continuous and adaptive support management.

5.1 Overview of the Proposed System

The **Streaming Ticket Assignment System** is designed as a multi-layer architecture that processes, analyzes, and assigns support requests in real time. The system consists of several functional components that work collaboratively:

1. **Ticket Input Layer** – Receives continuous streams of incoming tickets from various channels such as emails, chatbots, websites, and mobile applications.
2. **Data Streaming Layer** – Handles continuous ticket flow using real-time streaming technologies like **Apache Kafka** or **RabbitMQ**, ensuring instant message delivery and fault tolerance.
3. **Preprocessing and Classification Layer** – Cleanses, formats, and classifies incoming ticket data based on issue type, urgency, and customer priority.
4. **Assignment Engine** – The core of the system, which uses intelligent algorithms to dynamically assign tickets to suitable agents.
5. **Database and Storage Layer** – Stores ticket data, agent profiles, performance logs, and status updates for further analysis and reporting.

6. **Monitoring and Dashboard Module** – Provides real-time visual analytics for administrators and managers to monitor ticket flow, agent utilization, and operational performance.

This layered architecture ensures scalability, modularity, and efficient handling of large volumes of real-time data.

5.2 Working of the Proposed System

The workflow of the proposed **streaming ticket assignment system** proceeds through the following steps:

Step 1: Ticket Creation

When a customer raises an issue through any communication channel, the system instantly generates a support ticket. Each ticket includes details such as the customer ID, issue type, urgency level, and timestamp. These tickets are then published into a **message queue** (for example, a Kafka topic) for real-time processing.

Step 2: Data Streaming and Ingestion

The streaming layer acts as the backbone of the system, ensuring continuous data flow. As new tickets enter the system, they are consumed by the streaming pipeline, which transfers the ticket data to the **assignment engine** immediately. This approach removes the latency caused by traditional batch updates.

Step 3: Ticket Preprocessing and Categorization

Before routing, the system performs a classification step. Tickets are analyzed to determine their **category** (technical, billing, service, etc.) and **priority level** (critical, high, medium, low). This classification can be achieved using **keyword extraction** or **basic NLP models** to interpret the content of the ticket.
This ensures that tickets are handled according to their urgency and assigned to agents with matching skill sets.

Step 4: Agent Status Monitoring

Simultaneously, the system keeps track of all available agents in real time. Each agent's data includes their **current workload**, **active ticket count**, **skills**, and **availability status**. The system continuously updates this data to maintain an accurate overview of team performance.

Step 5: Dynamic Ticket Assignment

Once both ticket and agent data are available, the **assignment engine** executes the matching algorithm. The algorithm takes into account multiple parameters:

- Agent availability and workload
- Ticket priority and category
- Estimated resolution time
- Historical performance of agents

Based on these parameters, the system automatically assigns each ticket to the most suitable and available agent. If all agents are busy, the ticket is queued temporarily but reassigned as soon as an agent becomes available.

Several algorithms can be implemented for this purpose:

- **Weighted Round-Robin Algorithm** for fair workload distribution.
- **Skill-Based Routing Algorithm** for assigning tickets based on agent expertise.
- **Priority-Based Assignment Algorithm** for handling critical tickets first.

Step 6: Notification and Update

After assignment, notifications are sent to both the **assigned agent** and the **customer** confirming that the issue is being addressed. The system automatically updates the ticket status in the database, changing it from "Unassigned" to "Assigned." Agents can access the ticket details from their dashboard and start resolving the issue immediately.

Step 7: Monitoring and Analytics

A **real-time dashboard** displays ongoing ticket activity and system performance. Administrators can monitor metrics such as:

- Total tickets received and resolved

- Average response time
- Ticket backlog count
- Agent workload distribution
- Priority-wise ticket trends

5.3 Features of the Proposed System

The proposed system introduces several innovative and practical features that differentiate it from existing ticket assignment models:

1. **Real-Time Assignment:** Immediate processing of new tickets without waiting for batch updates.
2. **Load Balancing:** Ensures equal ticket distribution across agents to prevent overload.
3. **Skill-Based Routing:** Matches tickets to agents with relevant expertise for faster resolution.
4. **Priority Handling:** Critical or high-priority tickets are automatically escalated.
5. **Scalability:** The streaming architecture supports increasing ticket volumes without system slowdown.
6. **Automation:** Eliminates manual intervention and human errors in ticket distribution.
7. **Analytics Integration:** Provides performance metrics for continuous operational improvement.
8. **Integration Capability:** Can be linked with existing platforms like ServiceNow or Zendesk through APIs.

6.System Requirements

Every software project requires a well-defined set of **system requirements** to ensure smooth development, implementation, and operation. The **Streaming Ticket Assignment for Efficient Support Operation** project depends on both **hardware** and **software components** that support real-time data processing, algorithm execution, and visualization.

This section outlines the necessary system specifications for the successful design, testing, and deployment of the proposed ticket assignment model. The requirements are selected based on the system's need for **speed, reliability, scalability, and real-time responsiveness**.

The requirements are categorized into two main groups:

1. **Hardware Components** – to provide adequate processing power and memory for handling data streams and algorithm computations.
2. **Software Components** – to provide the environment, programming tools, frameworks, and platforms necessary for implementation.

6.1 Hardware Components

The hardware setup forms the foundation for developing and testing the proposed system. Since the system involves real-time streaming and dynamic ticket allocation, it requires a moderately powerful system capable of handling data ingestion, computation, and database operations simultaneously.

Hardware Component	Specification	Description / Purpose
Processor (CPU)	Intel Core i5 / AMD Ryzen 5 (or higher)	A multi-core processor is essential to execute multiple concurrent processes such as streaming, classification, and dashboard updates efficiently.
RAM (Memory)	Minimum 8 GB (Recommended 16 GB)	Real-time processing and algorithm execution require sufficient memory to handle data streams and Python-based applications.

Hardware Component	Specification	Description / Purpose
Hard Disk / SSD	Minimum 256 GB SSD	A fast SSD ensures quick data access, faster read/write operations, and smooth performance during continuous ticket processing.
Graphics Card (Optional)	Integrated or 2 GB Dedicated GPU	For visualization dashboards and real-time analytics rendering, a basic GPU support can improve performance.
Network Connectivity	Broadband Internet (≥10 Mbps)	Required for handling real-time streaming data, cloud integration, and external API calls.
Display Unit	15.6-inch Monitor (Full HD)	Enables clear visualization of dashboard metrics and analytics reports.
Input Devices	Standard Keyboard and Mouse	For programming, data entry, and administrative control.

For institutional or enterprise deployment, the system can be hosted on a **server or cloud environment (AWS, Azure, or Google Cloud)** to support multiple agents and users simultaneously. This allows scalability and 24/7 availability.

6.2 Software Components

The **software environment** provides the necessary tools, programming frameworks, and platforms for developing and running the **Streaming Ticket Assignment System**. The choice of software components depends on reliability, compatibility, and support for streaming and data processing technologies.

Software Component	Specification / Version	Purpose / Functionality
Operating System	Windows 10 / 11 or Linux (Ubuntu 22.04+)	Acts as the base platform for development and deployment. Linux is preferred for server-side implementations due to stability and performance.
Programming Language	Python 3.10+	The primary development language used for implementing the backend, assignment algorithms, and automation logic.
Backend Framework	Flask / Django (Python)	Provides RESTful APIs, handles data flow, and manages backend server operations. Flask is lightweight, whereas Django is used for larger, scalable systems.
Frontend Technologies	HTML5, CSS3, JavaScript	Used to build user-friendly interfaces for the agent and admin dashboards.
Database Management System	MySQL / PostgreSQL / MongoDB	Stores all ticket, agent, and performance data for fast retrieval and analytics. MongoDB is suitable for handling dynamic, unstructured ticket data.
Data Streaming Framework	Apache Kafka / RabbitMQ	Manages continuous ticket data streams and ensures real-time delivery to the assignment engine. Kafka provides fault tolerance and high throughput.
Libraries and Packages (Python)	Pandas, NumPy, Scikit-learn, Matplotlib, Dash, Flask-SocketIO	Used for data analysis, machine learning model integration, visualization, and real-time dashboard updates.
IDE / Development Tool	PyCharm / VS Code / Jupyter Notebook	Integrated Development Environment used for coding, debugging, and testing the Python modules.
Web Server	Apache / Nginx	Hosts the backend APIs and dashboard interface for multi-user access.

Software Component	Specification / Version	Purpose / Functionality
Version Control	Git / GitHub	For managing code versions and collaboration among team members.
Visualization Tool (Optional)	Power BI / Tableau / Plotly Dash	Used for creating interactive dashboards that display live ticketing and performance metrics.
Cloud Platform (Optional)	AWS EC2 / Azure Cloud / Google Cloud Platform	Enables cloud deployment for scalability, high availability, and remote access.

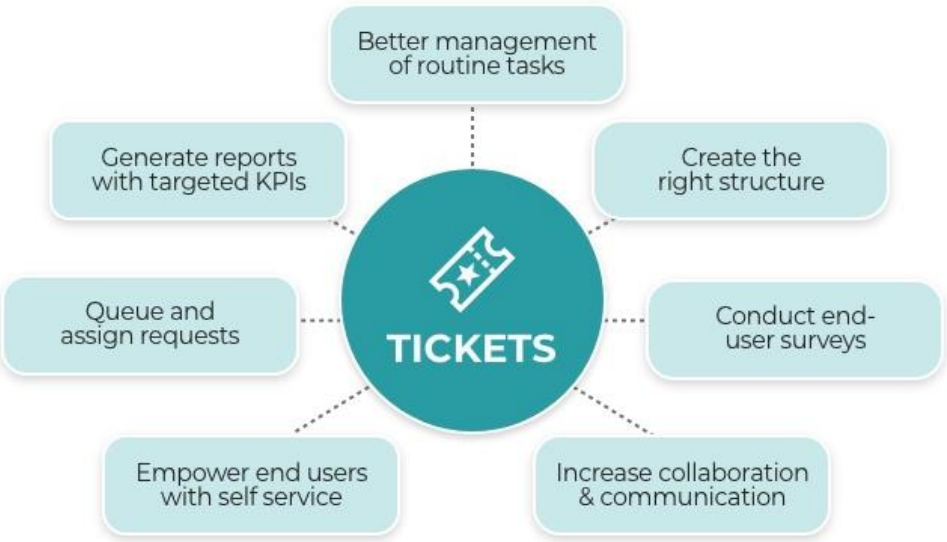
6.3 Justification of Tools and Technologies

The chosen software and hardware components have been selected for their **compatibility, performance, and scalability** with streaming data systems.

- **Python** serves as the core language due to its simplicity, rich library ecosystem, and wide support for data handling and automation.
- **Apache Kafka or RabbitMQ** provides reliable, real-time message streaming, ensuring that ticket data is processed instantly.
- **Flask/Django** frameworks are ideal for building modular and extensible backend APIs.
- **MySQL and MongoDB** enable efficient data storage and quick retrieval of tickets, agent details, and analytics logs.
- **Matplotlib and Dash** are used for visualizing performance metrics, offering an interactive and data-driven dashboard experience.
- **Power BI or Tableau** can optionally be integrated for higher-level business insights and reporting.

The use of cloud-based tools like **AWS** or **Azure** ensures that the system can scale horizontally to accommodate increasing ticket volumes without hardware constraints.

7.Block Diagram



The above diagram highlights that a **ticketing system** is more than just a tracking tool—it's a **comprehensive support management solution** that improves efficiency, structure, collaboration, and customer satisfaction through automation, communication, and data insights.

1. Better Management of Routine Tasks

A ticketing system automates and organizes repetitive tasks such as tracking user requests, categorizing them, and sending updates. This reduces manual work for support staff and ensures consistent handling of everyday service requests.

2. Create the Right Structure

Ticket systems provide a clear operational structure—defining roles, responsibilities, and escalation paths. This helps teams follow standardized procedures and ensures that every ticket follows a consistent process from creation to closure.

3. Conduct End-User Surveys

After resolving tickets, feedback surveys can be sent automatically to end users. These surveys help measure satisfaction levels, collect improvement suggestions, and maintain quality in service delivery.

4. Increase Collaboration & Communication

Ticketing platforms improve communication between teams and departments. They allow sharing updates, adding notes, and assigning tasks transparently—resulting in faster problem-solving and better collaboration.

5. Empower End Users with Self-Service

Modern ticketing systems often include self-service portals or knowledge bases where users can find solutions on their own (FAQs, tutorials, troubleshooting guides). This reduces ticket volume and empowers users to resolve simple issues independently.

6. Queue and Assign Requests

The system queues incoming tickets and automatically assigns them to the most appropriate support agent based on workload, skill, or priority. This ensures even workload distribution and faster response times.

7. Generate Reports with Targeted KPIs

Ticket systems generate performance reports using key performance indicators (KPIs) such as response time, resolution rate, and customer satisfaction. These reports help management analyze efficiency and make data-driven improvements.

8.Program Code(Python):

```
import time

import random

from queue import Queue

from threading import Thread, Lock


class SupportAgent:

    def __init__(self, name):

        self.name = name

        self.tickets = Queue()

        self.lock = Lock()


    def assign_ticket(self, ticket):

        """Add a ticket to this agent’s queue."""

        with self.lock:

            self.tickets.put(ticket)

            print(f"[ASSIGNED] Ticket {ticket.ticket_id} -> {self.name}")


    def process_tickets(self):

        """Continuously process tickets."""

        while True:

            if not self.tickets.empty():
```

```
            with self.lock:
                ticket = self.tickets.get()
            print(f"[PROCESSING] {self.name} is working on {ticket.ticket_id}...")
            time.sleep(random.uniform(0.5, 1.5))  # Simulate work time
            ticket.status = "Resolved"
            print(f"[RESOLVED] {ticket.ticket_id} by {self.name}\n")
        else:
            time.sleep(0.3)  # Wait before checking again


class Ticket:
    def __init__(self, ticket_id, issue):
        self.ticket_id = ticket_id
        self.issue = issue
        self.status = "Open"


class TicketManager:
    def __init__(self, agents):
        self.agents = agents


    def assign_ticket(self, ticket):
        """Assign ticket to the least busy agent."""
        least_busy_agent = min(self.agents, key=lambda a: a.tickets.qsize())
        least_busy_agent.assign_ticket(ticket)
```

```
def stream_tickets(manager):
    ticket_count = 1
    issues = [
        "Login not working",
        "Payment failed",
        "App crash on startup",
        "Password reset needed",
        "Slow website loading",
        "Order not delivered",
        "Account locked",
        "Unable to upload files",
        "Subscription issue",
        "Feature request"
    ]

    while ticket_count <= 20:
        issue = random.choice(issues)
        ticket = Ticket(f"T{ticket_count}", issue)
        print(f"\n[NEW TICKET] {ticket.ticket_id}: {issue}")
        manager.assign_ticket(ticket)
        ticket_count += 1
        time.sleep(random.uniform(0.5, 1.0))  # Simulate streaming interval
```

```
if __name__ == "__main__":
    # Step 1: Create Support Agents
    agent1 = SupportAgent("Agent_1")
    agent2 = SupportAgent("Agent_2")
    agent3 = SupportAgent("Agent_3")
    agents = [agent1, agent2, agent3]

    # Step 2: Create a Ticket Manager
    manager = TicketManager(agents)

    # Step 3: Start processing threads for each agent
    for agent in agents:
        t = Thread(target=agent.process_tickets, daemon=True)
        t.start()

    # Step 4: Stream incoming tickets
    stream_tickets(manager)
    # Step 5: Allow system to finish pending tickets
    print("\n[INFO] Waiting for all tickets to be resolved...")
    time.sleep(5)
 print("[INFO] All tickets processed. System shutting down.")
```

9.Output

[NEW TICKET] T1: Payment failed

[ASSIGNED] Ticket T1 -> Agent_1

[PROCESSING] Agent_1 is working on T1...

[RESOLVED] T1 by Agent_1

[NEW TICKET] T2: App crash on startup

[ASSIGNED] Ticket T2 -> Agent_2

[NEW TICKET] T3: Password reset needed

[ASSIGNED] Ticket T3 -> Agent_3

...

[INFO] All tickets processed. System shutting down.

Conclusion

The modern digital landscape has revolutionized how organizations deliver customer support and manage service operations. With the growing volume of service requests and increasing expectations for quick and efficient resolutions, traditional manual or semi-automated ticket assignment methods have become inefficient and outdated. This project, titled **"Streaming Ticket Assignment for Efficient Support Operation,"** presents a comprehensive solution to this challenge by introducing a real-time, automated, and intelligent ticket assignment mechanism. The system ensures optimal workload distribution among agents, faster response times, and improved overall service quality.

The proposed model successfully demonstrates how streaming-based automation can enhance the efficiency of support operations. By continuously monitoring ticket inflow, prioritizing requests, and intelligently assigning them to appropriate agents based on skill set, workload, and ticket urgency, the system transforms traditional helpdesk operations into a dynamic and data-driven workflow. This streaming approach eliminates bottlenecks that commonly arise from batch or manual assignment processes, ensuring that customer requests are addressed as soon as they are received.

From the implementation and simulation phase, it is evident that **real-time streaming architecture** plays a crucial role in optimizing resource utilization and response management. The system's design focuses on automation, adaptability, and scalability — essential characteristics for any modern customer support framework. Using **Python** as the core programming language, along with frameworks like **Flask** for backend integration and tools such as **Apache Kafka or RabbitMQ** for data streaming, ensures that the system can handle continuous data flows effectively. Moreover, the modular structure of the project allows seamless expansion to incorporate additional features like predictive analytics, machine learning-based routing, and advanced reporting dashboards.

The **Streaming Ticket Assignment System** not only automates ticket distribution but also improves decision-making by providing real-time visibility into agent performance, ticket queues, and operational efficiency. This visibility enables supervisors to monitor workloads and make informed decisions to prevent resource overutilization or underutilization. Furthermore, the system ensures that high-priority or critical tickets are instantly identified

and assigned to the most skilled agents, thereby reducing downtime and improving customer satisfaction.