Project Design Phase
Designing the Solution Architecture

Date	15 February 2026
Team ID	LTVIP2026TMIDS89549
Project Name	Streamlining Ticket Assignment for Efficient Support Operations
Maximum Marks	4 Marks

Goals of the Architecture:

The primary objective of the proposed architecture is to establish a fully automated, secure, and efficient ticket management framework within the **ServiceNow** platform. The solution architecture aims to:

- **Enhance Operational Efficiency:**
  Eliminate manual intervention in ticket routing and assignment, allowing support teams to focus on problem resolution rather than administrative tasks.
- **Optimize Resource Utilization:**
  Ensure balanced workload distribution among different support groups by automatically assigning tickets to the most suitable teams or agents based on predefined logic.
- **Reduce Issue Resolution Delays:**
  Minimize time spent in ticket triage and reassignment by ensuring accurate routing upon ticket creation.
- **Promote Accountability and Transparency:**
  Maintain clear traceability of ticket ownership and activity logs, ensuring smooth handover between teams.
- **Enable Scalability:**
  Provide a flexible architecture that can be easily expanded to accommodate new modules such as Change Requests or Problem Management in the future.

Key Components of the Architecture:

1. **Operations Related Table (u_operations_related)**

   - This is a **custom table** created to store ticket information such as *issue type, priority, description, assigned group,* and *status*.
   - It acts as the **central repository** for all support requests and serves as the trigger point for automation workflows.
   - Each record in this table represents a new support ticket generated by the user or administrator.

2. **sys_user_group Table**

   - A native **ServiceNow** table used to store details of support groups such as *Certificates Team* and *Platform Team*.

- It ensures that users are properly categorized according to their responsibilities and access levels.
- These groups form the **target entities** for automated ticket routing.

3. **Flow Designer**

    - The **Flow Designer** is used to implement two distinct automated flows that handle ticket assignment logic.
    - Each flow monitors the *'Issue'* field in the Operations Related table and automatically updates the *'Assigned_to_group'* field based on the selected issue category.
    - This no-code/low-code approach simplifies automation without requiring external scripts or integrations.

4. **Flow Logic**

    - The logic includes **conditional checks** that evaluate the value in the *'Issue'* choice field.
    - For example:
        - If *Issue = Certificate Access*, assign to *Certificates Group*.
        - If *Issue = Platform Error*, assign to *Platform Group*.
    - This ensures each issue is directed to the appropriate group for resolution, reducing the chance of misrouting.

5. **Access Controls (ACLs)**

    - Access Control Lists are configured to protect the newly created Operations Related table.
    - They restrict modification, deletion, or viewing of tickets based on user roles.
    - This guarantees **data confidentiality and operational security**, preventing unauthorized access or accidental data loss.

Development Phases:

The architecture was implemented in **six structured stages**, ensuring smooth deployment and validation:

1. **Creation of Test Data:**
   Test users such as *Katherine* and *Manne* were created and assigned to appropriate groups and roles to simulate real-world support operations.
2. **Custom Table Configuration:**
   The *Operations Related* table was designed with essential fields like *Issue, Priority, Assigned_to_group,* and *Description*.
   Choice fields were added to represent different types of issues that would determine routing behavior.

3. **Security and Role Setup:**
   Roles were defined for different user categories (Admin, Support Agent, Manager). ACLs were configured to ensure that only authorized roles could create, read, or update records in the custom table.

4. **Flow Implementation in Flow Designer:** Two automated flows were created:

   - One for newly created tickets.
   - Another for updated records.
     Each flow checks the *'Issue'* field and updates the *'Assigned_to_group'* accordingly.

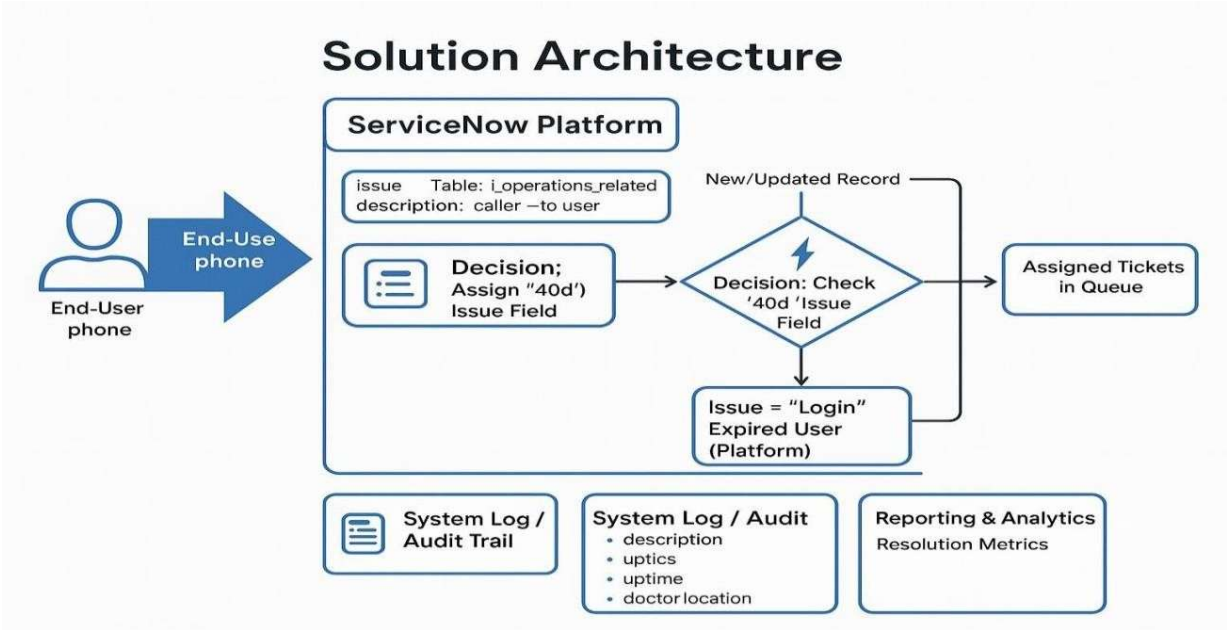
5. **Access Control and Flow Validation:**
   The flow logic and ACLs were tested together to ensure there were no conflicts. Each issue type was verified to confirm correct routing and group assignment.

6. **Testing and Verification:**
   Sample tickets were generated to test the system's accuracy.
   The automation successfully assigned tickets to their respective groups without manual intervention, confirming functionality and stability.

Solution Architecture Description:

The proposed solution architecture for **Streamlining Ticket Assignment** is designed to automate and optimize ticket routing in **ServiceNow**, ensuring high accuracy and reduced resolution time.

Example - Solution Architecture Diagram:



The system revolves around a custom table called *Operations Related*, which acts as the core database for all ticket-related records. Whenever a new ticket is created, the **Flow Designer** triggers an automation that reads the *'Issue'* field. Based on predefined logic, the flow automatically determines which group should handle the issue and populates the

'Assigned_to_group' field. This process eliminates the need for manual ticket sorting and ensures that every issue reaches the correct team instantly.

Security is reinforced through **Access Control Lists (ACLs)** that define permissions for users and groups. These controls ensure that only authorized individuals can view or modify ticket data, maintaining integrity and confidentiality across all processes.

Additionally, the architecture's modular design allows for **future scalability**. The same automation framework can be easily adapted for other modules such as *Incident Management, Problem Management, Change Requests,* or *Service Catalog Items*. This adaptability ensures long-term sustainability and flexibility within the organization's IT Service Management (ITSM) ecosystem.

By integrating automated flows, structured data design, and robust access controls, this solution architecture significantly reduces manual workload, enhances service responsiveness, and supports a more proactive and data-driven approach to support operations at *ABC Corporation*.