Tailored Application Access for Enhanced User Experience

1. Project Overview

This project aims to create a custom ServiceNow table named "Auto-Populated" to capture service request information. The goal is to streamline the service request process by automating certain fields and controlling access to the table based on user roles.

2. Objectives

Create a Custom Table: Develop a new table to store service request details.

Implement Role-Based Access Control: Restrict access to the table and its actions to specific user roles.

Configure Table Layout: Define the table's columns and their data types.

Create User Roles and Groups: Establish user roles and groups to manage access permissions.

3. Key Features and Concepts Utilized

Custom Table: A tailored table to store specific service request data.

Role-Based Access Control: A security mechanism to control user access to the table and its actions.

Module Development: Creation of custom modules to provide a user-friendly interface for service request management

4. Detailed Steps to Solution Design

1**. Create a New Table**:

- Define the table's label ("Service Request") and name ("Auto-Populated").
- Add necessary columns: "Name" (String) and "Issue" (String).
- Set appropriate visibility and accessibility settings.

2. **Create User Roles and Groups**:

- Create a new user role, e.g., "Service Request Manager."
- Assign the "Service Request Manager" role to the desired user (Jai Prakash).
- Create a group, e.g., "Manager Group," and add the "Service Request Manager" role to it.

3. **Configure Role-Based Access Control**:

- Assign the "Service Request Manager" role to the "Auto-Populated" table.
- Configure the table's access controls to restrict operations to authorized users.

4. **Develop Custom Modules**:

- Create a "Create New" module to allow authorized users to create new service requests.
- Create an "All" module to display a list of all service requests.
- Configure module visibility based on user roles.

5. Testing and Validation

Unit Testing: Test individual components of the solution, such as script includes and business rules.

User Interface Testing: Validate the user interface to ensure it is intuitive and easy to use.

Functional Testing: Verify that the solution meets all functional requirements and performs as expected.

Security Testing: Assess the security measures to ensure that unauthorized users cannot access sensitive information.

6. Key Scenarios Addressed by ServiceNow in the Implementation Project

- **Controlled Access**: Restricting access to the "Auto-Populated" table to authorized users.
- **Simplified Service Request Creation:** Providing a user-friendly interface for creating new service requests.
- **Centralized Service Request Management:** Centralizing service request information in a dedicated table.

7. Conclusion

By implementing this solution, we have successfully created a custom ServiceNow table to streamline the service request process. The role-based access control ensures that only authorized users can create and manage service requests, enhancing security and efficiency.