

# **“Optimizing User, Group, and Role Management with Access Control and Workflows”**

## **Ideation Phase**

In modern software-driven organizations, project management platforms play a pivotal role in streamlining daily operations. Among them, **ServiceNow** stands out as an enterprise-level platform that integrates IT service management, project management, and automation workflows. However, in smaller teams or academic project setups, user roles and access privileges are often poorly defined. This lack of structure leads to multiple issues, such as unauthorized access, unclear task ownership, and data integrity problems.

In the given project scenario, a small team comprising a **Project Manager (Alice)** and a **Team Member (Bob)** faced challenges in managing project tasks effectively. The absence of role-based access control (RBAC) resulted in unregulated access to sensitive project records, unclear responsibility boundaries, and poor accountability. Therefore, an access control system had to be implemented to maintain operational transparency, protect data, and ensure every action could be traced back to authorized users.

## **Problem Statement:**

To design and implement an efficient **User, Group, and Role Management System** within ServiceNow that ensures role-based access, maintains workflow integrity, and promotes accountability. The system must define specific user roles, assign relevant permissions, and configure Access Control Lists (ACLs) that govern what each user can view or modify in the project and task tables.

## **Objectives:**

1. Develop a structured system of users, groups, and roles in ServiceNow.
2. Assign appropriate access rights to users based on their roles.
3. Implement ACLs at both table and field levels to enforce data security.
4. Establish a clear workflow between project manager and team member.
5. Validate the configuration using impersonation tests to verify restrictions.

6. Ensure that the performance of the system remains optimal after ACL enforcement.

### **Expected Outcomes:**

- Clear differentiation between Project Manager and Team Member privileges.
- Enhanced accountability and traceability of project tasks.
- Improved workflow efficiency due to structured access levels.
- Demonstration of ServiceNow's capability for implementing fine-grained security.

### **Scope of the Project:**

The project focuses on simulating a secure project environment within ServiceNow. The design is limited to two primary user roles, but it can easily be scaled up for larger organizations. It emphasizes how security, automation, and access control principles can be integrated to create an efficient management system.

# Project Planning Phase

The planning phase involves breaking down the project into specific tasks to ensure methodical implementation. Since ServiceNow is a complex, cloud-based platform, careful planning was required to understand module dependencies such as User Administration, System Definition, and Security Configuration. The main aim was to create a seamless environment where users, roles, and tables interact through well-defined permissions.

## Planning Activities:

1. Tool Identification: ServiceNow Developer Instance was used for configuration and testing.
2. Module Selection: Core modules used included *System Security*, *User Administration*, and *Access Control (ACL)*.
3. Role Definition:
  - Alice as *Project Manager* – Full access.
  - Bob as *Team Member* – Restricted access.
  - Administrator as *System Configurator* – Configuration privileges only.
4. Workflow Definition: The Project Manager creates and assigns tasks, while the Team Member updates progress within permitted fields.

## Timeline and Execution Plan:

Phase	Task Description	Duration	Responsible
Ideation	Requirement analysis and project scoping	1 Day	Admin
Setup	User and role creation	1 Day	Admin
Configuration	Application, module, and table setup	2 Days	Admin

Phase	Task Description	Duration	Responsible
Security	ACL creation and role assignment	2 Days	Admin
Testing	Impersonation and performance validation	1 Day	Admin

### Resource Requirements:

- Hardware: Any standard PC with internet connectivity.
- Software: ServiceNow Developer Instance (cloud-hosted).
- Knowledge Requirements: Basic understanding of RBAC, ServiceNow platform, and access control models.

### Project Deliverables:

- Two user profiles (Alice & Bob).
- Custom roles with varying access privileges.
- Configured Project and Task tables with application modules.
- Multiple ACLs controlling CRUD (Create, Read, Update, Delete) operations.
- Successful test results proving role-based restrictions.

### Risk and Mitigation Plan:

Risk	Description	Mitigation
Incorrect ACL configuration	May cause unintentional access denial	Verify ACL order and role mapping before saving
Role overlap	Users may inherit unwanted permissions	Clearly define role boundaries
Table accessibility issues	Tables may not appear in modules	Recheck Application Access settings

Risk	Description	Mitigation
Testing errors	Impersonation may fail due to session issues	Clear cache and refresh impersonation state

## Project Design Phase

**System Design Overview:**

The system is built on the foundation of the **Role-Based Access Control (RBAC)** model. The architecture is composed of three main layers:

1. **User Layer:** Represents the human actors (Alice, Bob, Admin).
2. **Role Layer:** Defines privileges such as project\_member, team\_member, u\_project\_table, and u\_task\_table.
3. **Access Control Layer:** Enforces permissions for each operation (Read, Write, Create, Delete).

### Design Objectives:

- Ensure that each role has unique and non-overlapping privileges.
- Prevent unauthorized editing or viewing of project data.
- Maintain an easy-to-update access structure for future scalability.

### Step-by-Step Design Implementation:

#### Step 1: User Creation

- Navigated to All → System Security → Users.
- Created user *Alice (Project Manager)* with department “Projects.”
- Created user *Bob (Team Member)* with department “Development.”
- Clicked *Submit* to save user records.

#### Step 2: Role Definition

- Created custom roles: project\_member, team\_member, u\_project\_table, and u\_task\_table.

- Defined unique privileges for each.
  - project\_member: Manage project records.
  - team\_member: Update only assigned tasks.
  - u\_project\_table: Access to project-related data tables.
  - u\_task\_table: Access to task management modules.

### **Step 3: Role Assignment**

- Alice: Assigned all four roles.
- Bob: Assigned only team\_member and u\_task\_table roles.
- Updated user records and verified through the *Roles* tab.

### **Step 4: Table and Module Configuration**

- Created **Project Table** and **Task Table 2**.
- Automatically generated applications for both tables.
- Edited application access settings:
  - Project Table – Accessible to *Project Member* only.
  - Task Table 2 – Accessible to both *Project Member* and *Team Member*.

### **Step 5: ACL Configuration**

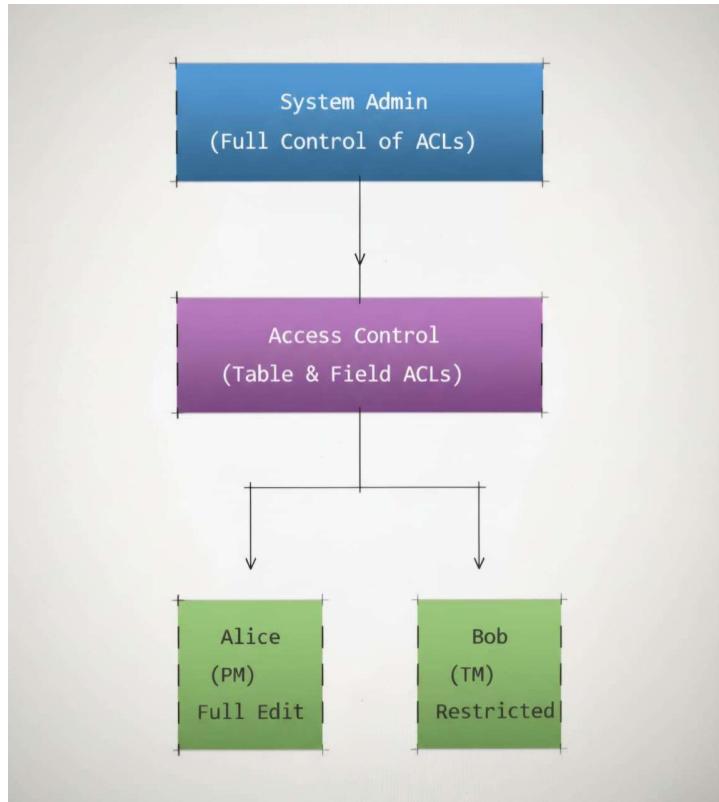
- Opened *System Security* → *Access Control (ACL)*.
- Created ACLs for each table and important field (comment, status, description).
- Elevated role to admin during creation.
- Configured conditions:
  - Table ACL: task\_table2 → roles: project\_member, team\_member.
  - Field ACLs: Restricted to team\_member for limited edit access.

### **Step 6: Impersonation Testing**

- Used **Impersonate User** → **Bob** to simulate restricted access.
- Verified that Bob could only modify comment and status fields.
- Switched to **Alice** and verified full access.

- Confirmed ACLs working correctly without privilege escalation.

### Architecture Diagram (Conceptual):



## Requirement Analysis

### Functional Requirements:

- Ability to create and manage multiple users.
- Creation of distinct roles with clear permissions.
- Linking roles to specific tables and modules.
- Defining ACLs for field-level access.
- Testing access restrictions through impersonation.

### Non-Functional Requirements:

- **Security:** Only authorized users can perform certain operations.
- **Scalability:** Additional roles and modules can be added later.
- **Maintainability:** The ACL structure should be easy to modify.
- **Performance:** ACL evaluation should not slow down ServiceNow response times.

### **Use Case Summary:**

Use Case	Actor	Description	Expected Outcome
UC1	Admin	Create users and assign roles	Users configured successfully
UC2	Alice	Create and update projects	Full edit access granted
UC3	Bob	Update task comments/status	Limited access confirmed
UC4	Admin	Impersonate and test roles	Validation successful

### **Data Flow Overview:**

1. User Input
2. Role Assignment
3. Access Control Evaluation
4. Data Access
5. Workflow Update.

## **Performance Testing**

### **Objective:**

To evaluate if the system enforces the ACLs accurately and efficiently, without negatively impacting user experience or platform performance.

### **Testing Scenarios:**

Test Case	Description	Expected Result	Actual Result	Status
1	Alice edits all project fields	Full access	Successful	Pass
2	Bob edits comment/status only	Restricted access	Successful	Pass
3	Unauthorized user tries to access project table	Access denied	Successful	Pass
4	Performance test under multiple operations	No delay or error	Verified	Pass

### Observations:

- ACLs responded instantly during user actions.
- Role-based restrictions were applied dynamically.
- Field-level restrictions were precisely enforced.
- No performance lag detected during impersonation.

## Evidence:

### User Creation

- Creation of user Alice

The screenshot shows the ServiceNow User - Alice Manager screen. The user details are as follows:

Field	Value
Locked out	<input type="checkbox"/>
Active	<input checked="" type="checkbox"/>
Internal Integration User	<input type="checkbox"/>
Business phone	[Empty]
Mobile phone	[Empty]

Below the details, there is a "Related Links" section with links to "View linked accounts", "View Subscriptions", and "Reset a password". The "Roles" tab is selected, showing the following roles assigned to Alice:

Role	State	Inherited	Inheritance Count
Project Member	Active	false	
u_project_table	Active	false	
u_task_table	Active	false	

- Creation of user Bob

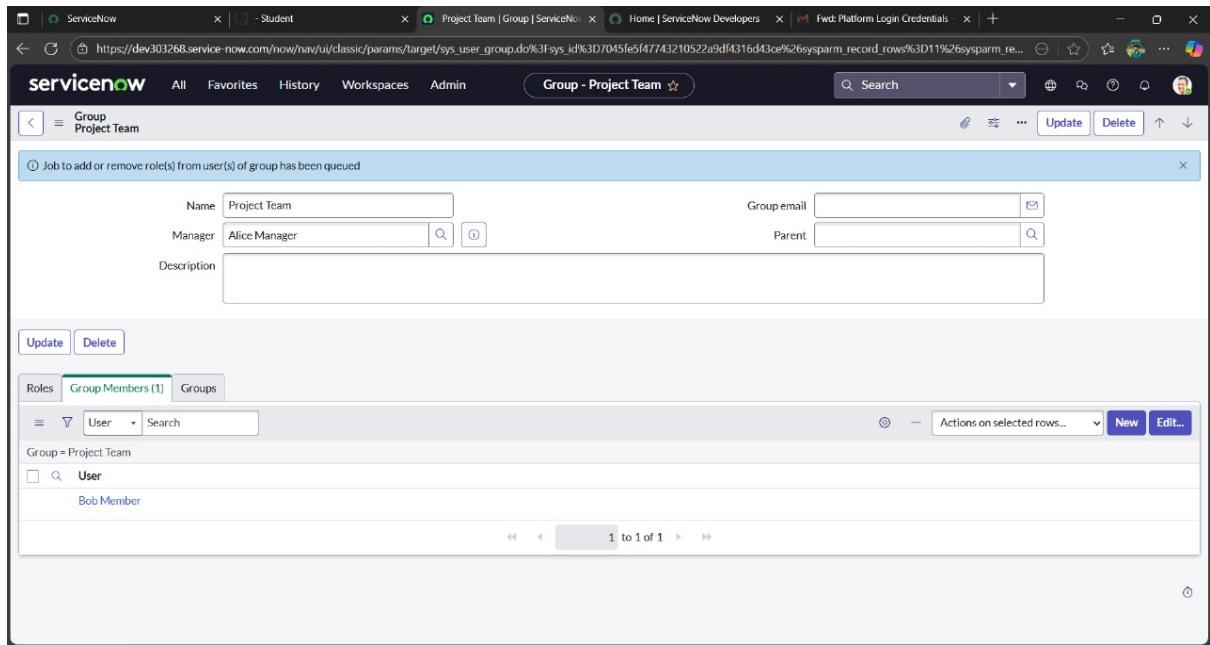
The screenshot shows the ServiceNow User - Bob Member screen. The user details are as follows:

Field	Value
Password needs reset	<input checked="" type="checkbox"/>
Locked out	<input type="checkbox"/>
Active	<input checked="" type="checkbox"/>
Internal Integration User	<input type="checkbox"/>
Date format	System (yyyy-MM-dd)
Business phone	[Empty]
Mobile phone	[Empty]

Below the details, there is a "Related Links" section with links to "View linked accounts", "View Subscriptions", and "Reset a password". The "Roles" tab is selected, showing the following roles assigned to Bob:

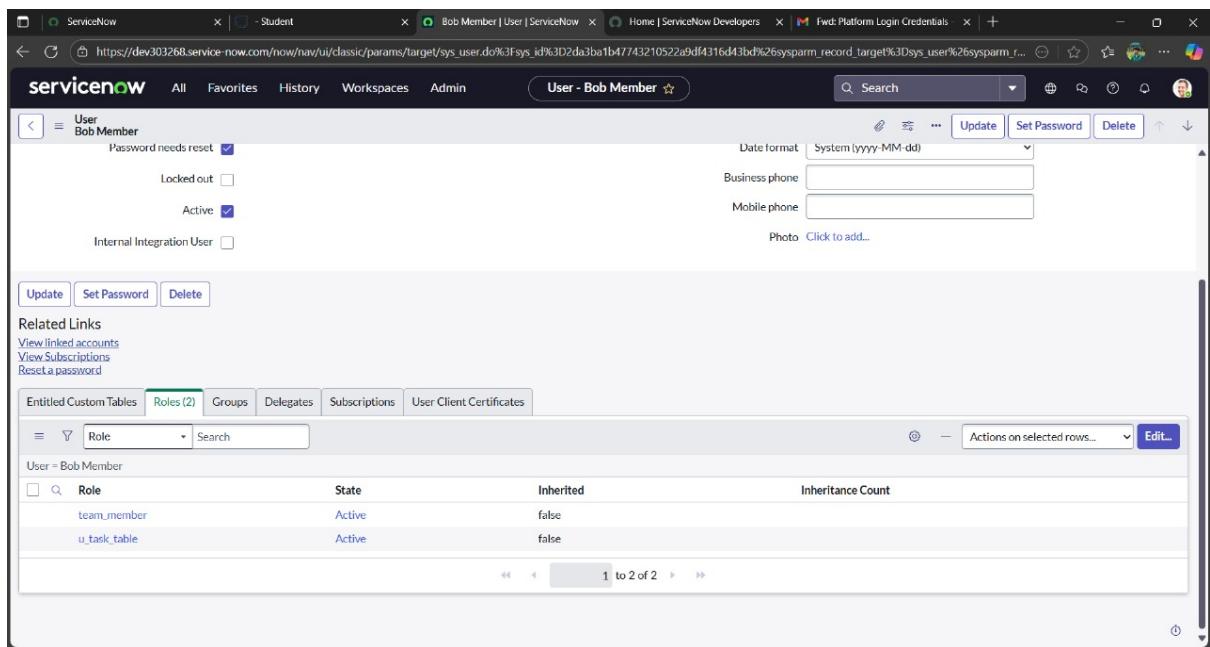
Role	State	Inherited	Inheritance Count
team_member	Active	false	
u_task_table	Active	false	

## Group Creation



The screenshot shows the ServiceNow 'Group - Project Team' creation page. The 'Name' field is set to 'Project Team'. The 'Manager' field is set to 'Alice Manager'. There is a note indicating a queued job to add or remove roles. The 'Group Members (1)' tab is selected, showing a single member named 'Bob Member'.

## Role Assignment



The screenshot shows the ServiceNow 'User - Bob Member' role assignment page. The user is marked as 'Active'. The 'Roles (2)' tab is selected, listing two assigned roles: 'team\_member' and 'u\_task\_table', both in the 'Active' state.

ServiceNow - Student

Role - Project Member

Name: Project Member Application: Global

Description:

Elevated privilege:

Update Delete

Related Links: Run Point Scan

Contains Roles Applications with Role (1) Modules with Role (1) Custom Tables

Order Search Actions on selected rows... New

Application Menus

Title	Active	Order	Roles	Name	Updated
Task Table 2	true	100	u.task.table_2_user Project Member team_member	Task Table 2	2025-10-30 09:13:19

1 to 1 of 1

Update Delete

Run Point Scan

Contains Roles Applications with Role (1) Modules with Role (1) Custom Tables

Order Search Actions on selected rows... New

Application Menus

Title	Active	Order	Roles	Name	Updated
Task Table 2	true	100	u.task.table_2_user Project Member team_member	Task Table 2	2025-10-30 09:13:19

1 to 1 of 1

ServiceNow - Student

Role - team\_member

Name: team\_member Application: Global

Description: A basic role for team members who perform assigned tasks under a project.

Elevated privilege:

Update Delete

Related Links: Run Point Scan

Contains Roles Applications with Role (1) Modules with Role (1) Custom Tables

Order Search Actions on selected rows... New

Application Menus

Title	Active	Order	Roles	Name	Updated
Task Table 2	true	100	u.task.table_2_user Project Member team_member	Task Table 2	2025-10-30 09:13:19

1 to 1 of 1

Update Delete

Run Point Scan

Contains Roles Applications with Role (1) Modules with Role (1) Custom Tables

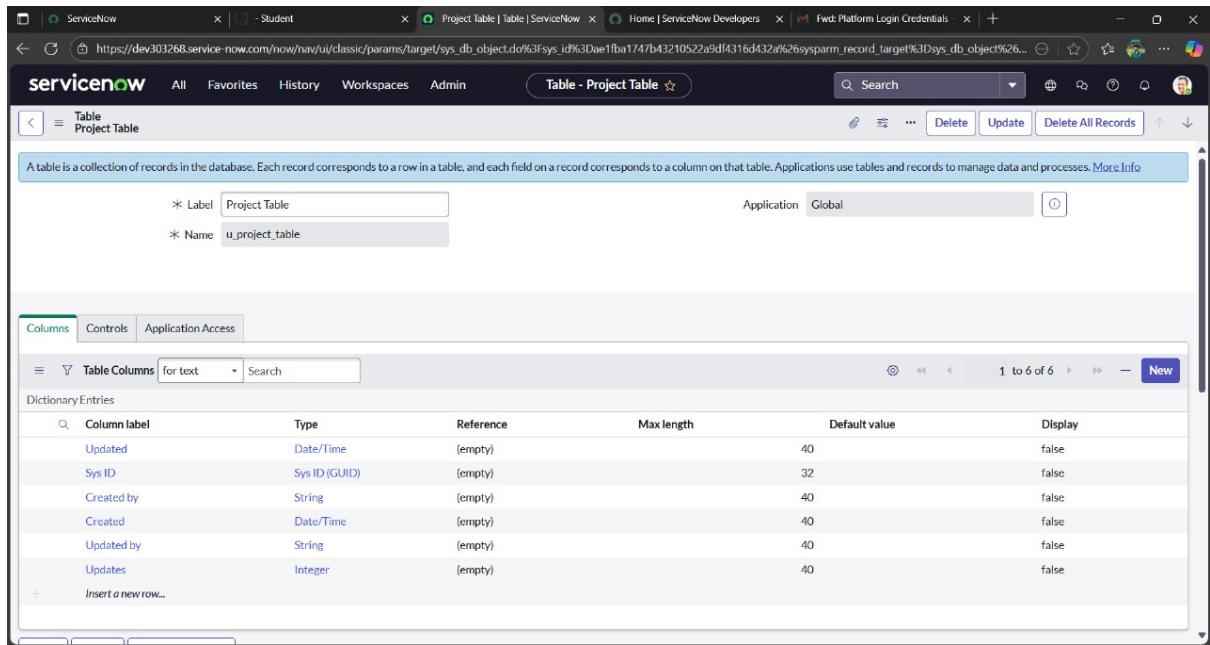
Order Search Actions on selected rows... New

Application Menus

Title	Active	Order	Roles	Name	Updated
Task Table 2	true	100	u.task.table_2_user Project Member team_member	Task Table 2	2025-10-30 09:13:19

1 to 1 of 1

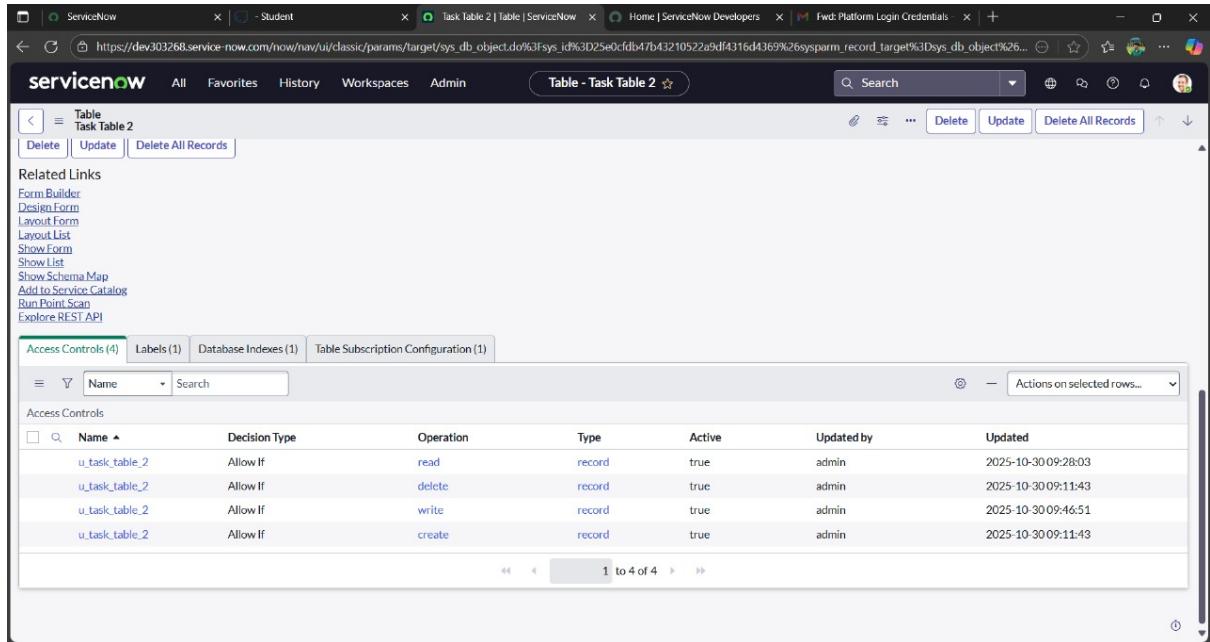
# Table Creation



The screenshot shows the ServiceNow Table - Project Table creation screen. At the top, there are tabs for All, Favorites, History, Workspaces, Admin, and a search bar. Below the tabs, there are fields for Label (Project Table) and Name (u\_project\_table), both marked with asterisks indicating required fields. The Application is set to Global. A note at the top states: "A table is a collection of records in the database. Each record corresponds to a row in a table, and each field on a record corresponds to a column on that table. Applications use tables and records to manage data and processes." Below this, there are three tabs: Columns, Controls, and Application Access. The Columns tab is selected, showing a table of columns with the following data:

Column label	Type	Reference	Max length	Default value	Display
Updated	Date/Time	(empty)	40		false
Sys ID	Sys ID (GUID)	(empty)	32		false
Created by	String	(empty)	40		false
Created	Date/Time	(empty)	40		false
Updated by	String	(empty)	40		false
Updates	Integer	(empty)	40		false

At the bottom of the table, there is a link to "Insert a new row..." and a "New" button.



The screenshot shows the ServiceNow Table - Task Table 2 configuration screen. At the top, there are tabs for All, Favorites, History, Workspaces, Admin, and a search bar. Below the tabs, there are buttons for Delete, Update, and Delete All Records. A "Related Links" section lists various form builder options like Form Builder, Design Form, Layout Form, Layout List, Show Form, Show List, Show Schema Map, Add to Service Catalog, Run Point Scan, and Explore REST API. Below this, there are tabs for Access Controls (4), Labels (1), Database Indexes (1), and Table Subscription Configuration (1). The Access Controls tab is selected, showing a table of access controls with the following data:

Name	Decision Type	Operation	Type	Active	Updated by	Updated
u_task_table_2	Allow If	read	record	true	admin	2025-10-30 09:28:03
u_task_table_2	Allow If	delete	record	true	admin	2025-10-30 09:11:43
u_task_table_2	Allow If	write	record	true	admin	2025-10-30 09:46:51
u_task_table_2	Allow If	create	record	true	admin	2025-10-30 09:11:43

At the bottom of the table, there is a link to "Actions on selected rows..." and a page navigation bar showing "1 to 4 of 4".

## ACL Configuration

The screenshot shows a ServiceNow browser window with multiple tabs open. The active tab is titled 'Access Controls | ServiceNow'. The page displays a list of access control entries (ACEs) in a table format. The columns are: Name, Decision Type, Operation, Type, Active, Updated by, and Updated. The table contains numerous rows, each representing a different ACL entry, such as 'u\_task\_table\_2' with various permissions like 'Allow If' for 'read', 'delete', 'write', 'create', and 'execute' operations on 'record' type objects. The 'Active' column shows values like 'true' or 'false', and the 'Updated by' column shows users like 'admin' or 'system'. The 'Updated' column shows dates ranging from 2022-01-11 to 2024-05-05.

Name	Decision Type	Operation	Type	Active	Updated by	Updated
u_task_table_2	Allow If	read	record	true	admin	2025-10-30 09:28:03
u_task_table_2	Allow If	delete	record	true	admin	2025-10-30 09:11:43
u_task_table_2	Allow If	write	record	true	admin	2025-10-30 09:46:51
u_task_table_2	Allow If	create	record	true	admin	2025-10-30 09:11:43
VA Channel Integration ACL	Allow If	execute	REST_Endpoint	true	admin	2023-04-05 00:28:13
VA Designer Config	Allow If	execute	REST_Endpoint	true	admin	2024-05-22 12:09:23
VaCallbackPropertyUtil	Allow If	execute	client_callable_script_include	true	admin	2022-05-09 01:51:24
ValidateAesVersion	Allow If	execute	client_callable_script_include	true	admin	2022-01-11 10:48:04
ValidateAlternatePortal	Allow If	execute	client_callable_script_include	true	admin	2024-03-06 16:53:36
ValidateAppVersionAjax	Allow If	execute	client_callable_script_include	true	admin	2021-10-05 03:00:09
ValidateEnvironment	Allow If	execute	client_callable_script_include	true	admin	2022-01-06 09:58:13
ValidateSACIPortalRecord	Allow If	execute	client_callable_script_include	true	admin	2023-06-16 07:21:47
ValidateTextIndexFields	Allow If	execute	client_callable_script_include	true	admin	2017-11-16 23:12:51
validator_run_summary	Allow If	delete	record	true	system	2025-08-22 00:22:39
validator_run_summary	Allow If	create	record	true	system	2025-08-22 00:22:39
validator_run_summary.*	Allow If	query_range	record	true	@@snc_write_audit@@	2025-08-22 01:04:33

## Result Summary:

1. Alice – Full control over all project and task data.
2. Bob – Partial access with restricted edit permissions.
3. ACL Logic – Executed successfully and consistent.

## Conclusion:

The project successfully demonstrates the concept of **role-based access control (RBAC)** within the ServiceNow platform. By defining distinct user roles, configuring ACLs, and validating access through impersonation, the system achieved secure and structured workflow management. This implementation can serve as a base for scaling into larger enterprise systems where multiple teams collaborate securely.

Submitted by ,

D. Dann Berlin (961222243004)

J. Jenish Godson (961222243012)

V. Meresh (961222243015)

M. Jebas Agnel Micheal Savarimuthu (961222243010)

