# “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**

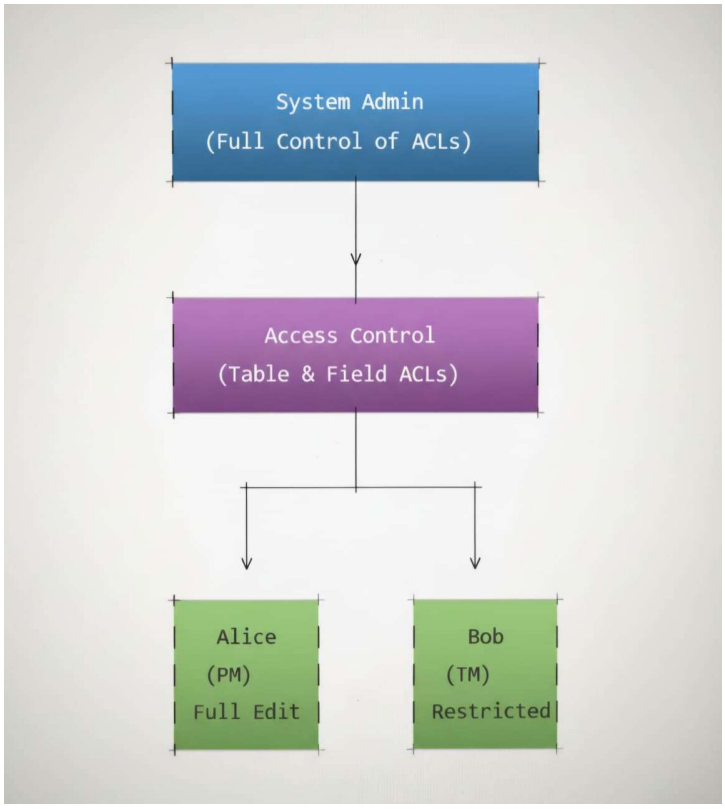
- Opened *System Security → Access Control (ACL)*.
- Created ACLs for each table and important field (comment, status, description).
- Elevated role to admin during creation.
- Configured conditions:
    - Table ACL: task_table2 → roles: project_member, team_member.
    - Field ACLs: Restricted to team_member 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 |

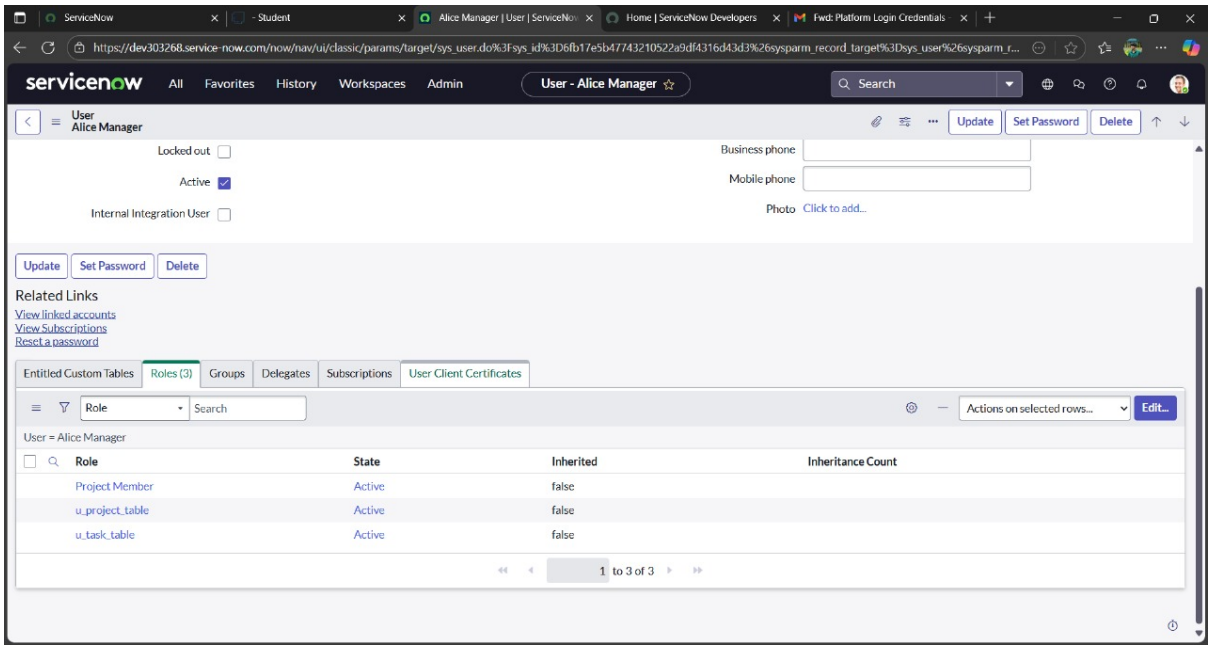
**Observations:**

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

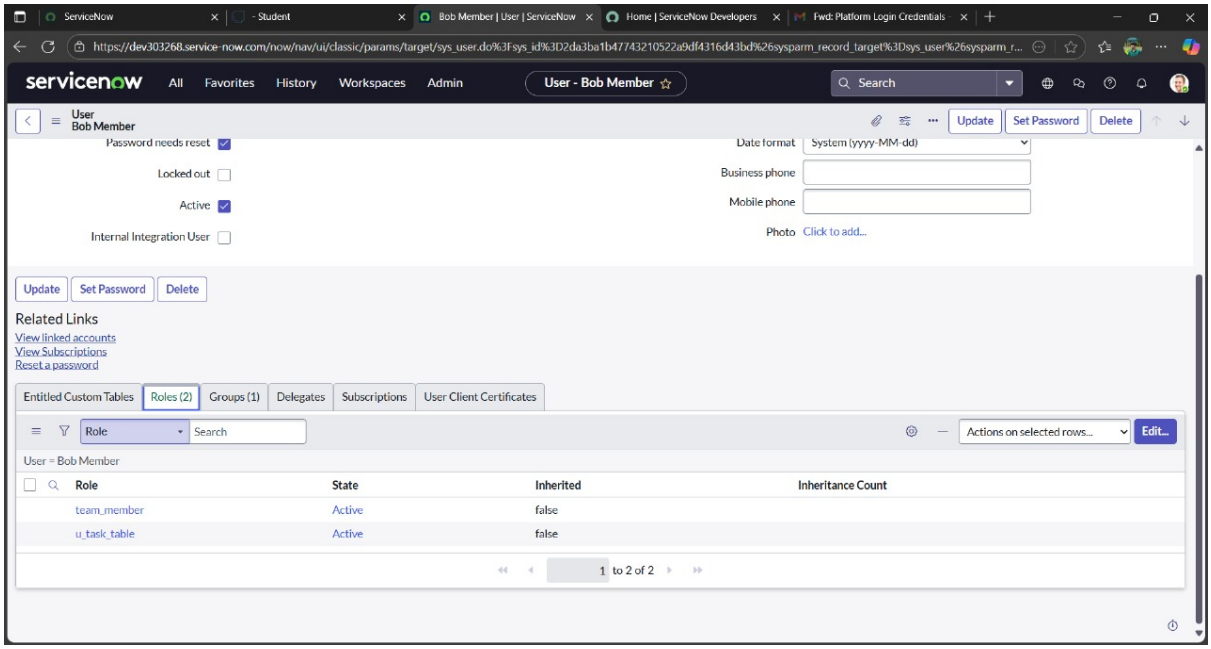
# Evidence:

## User Creation

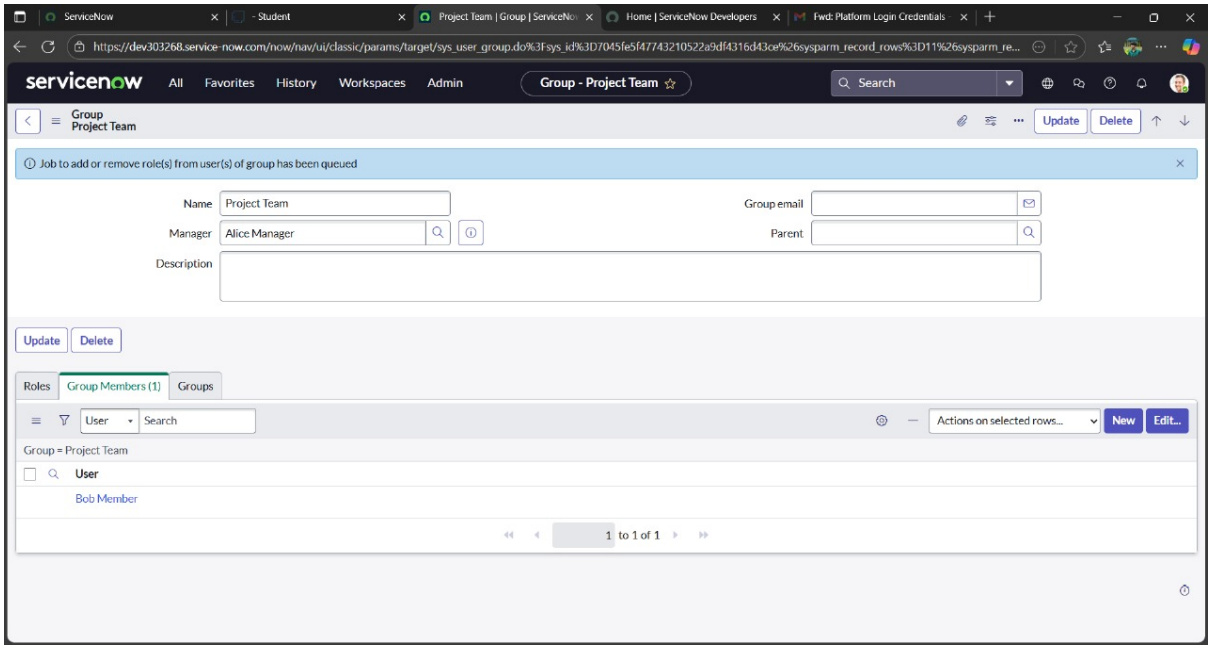
- Creation of user Alice



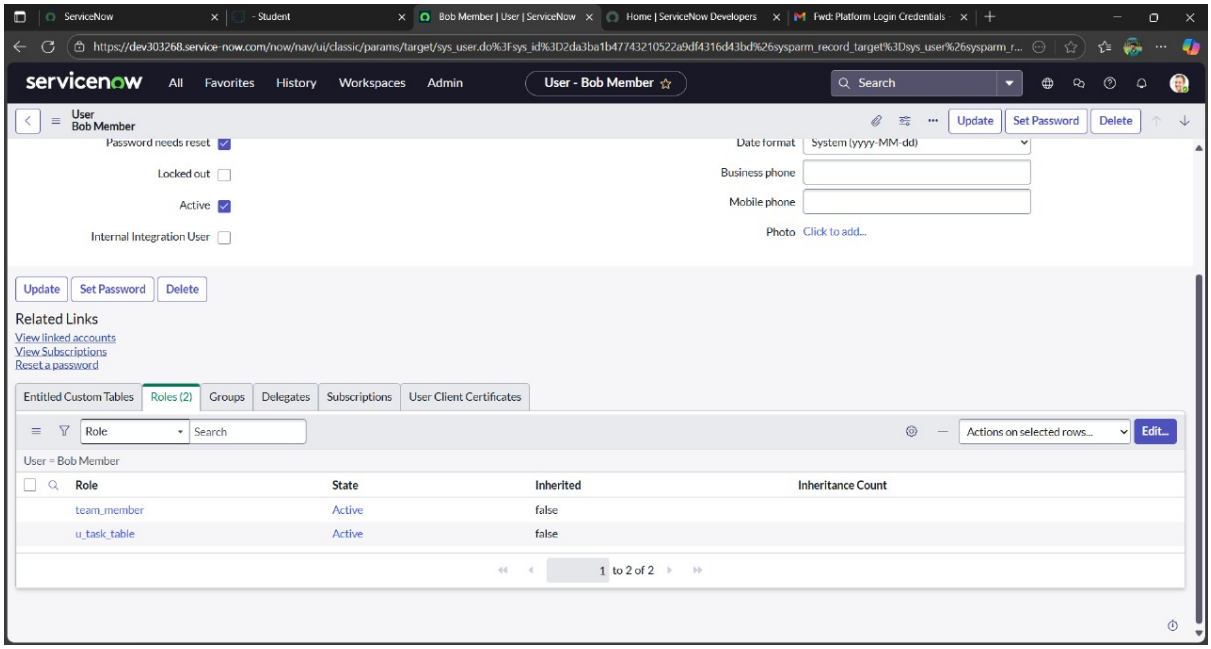

- Creation of user Bob

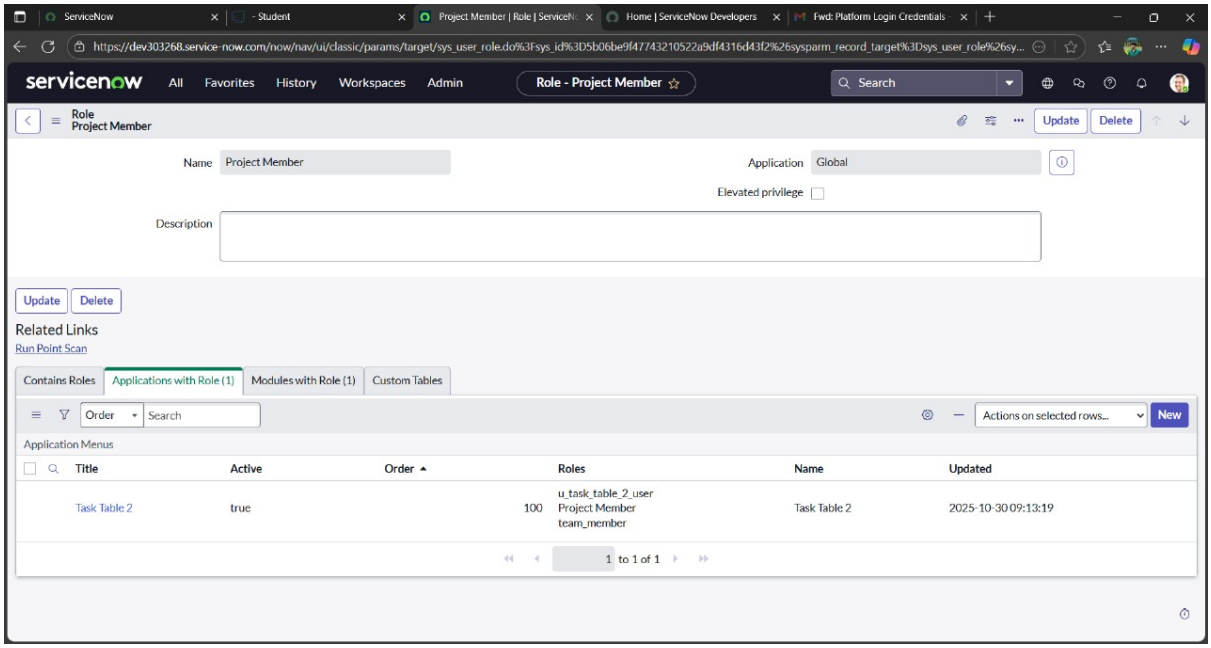


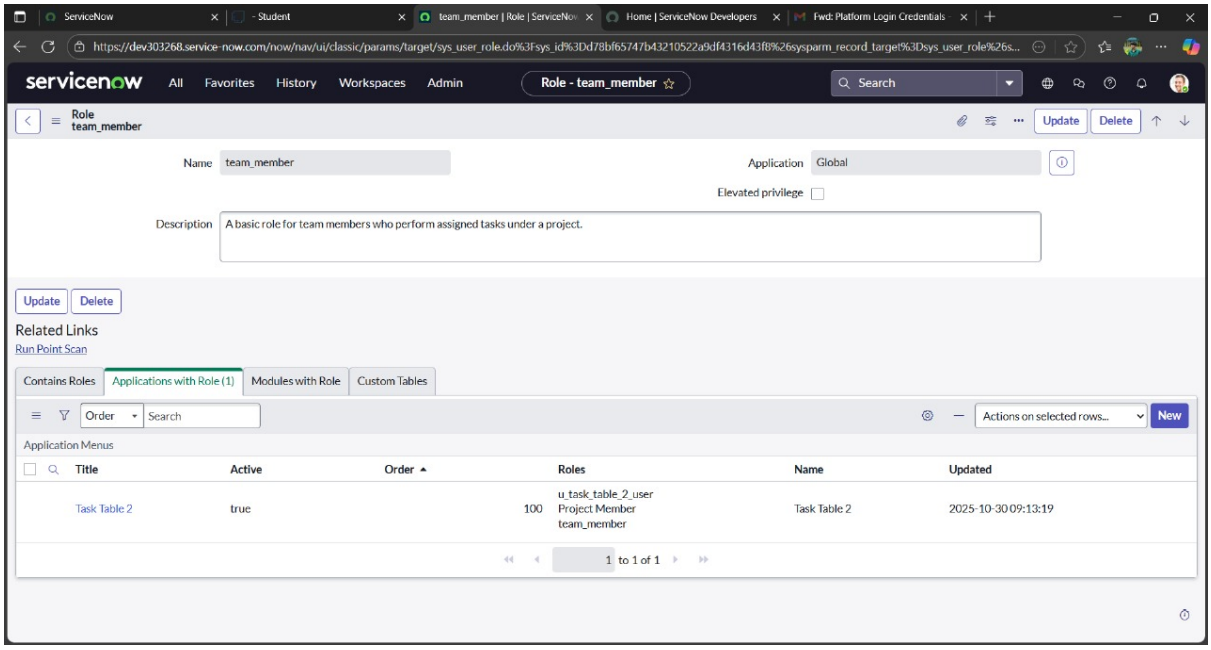
# Group Creation



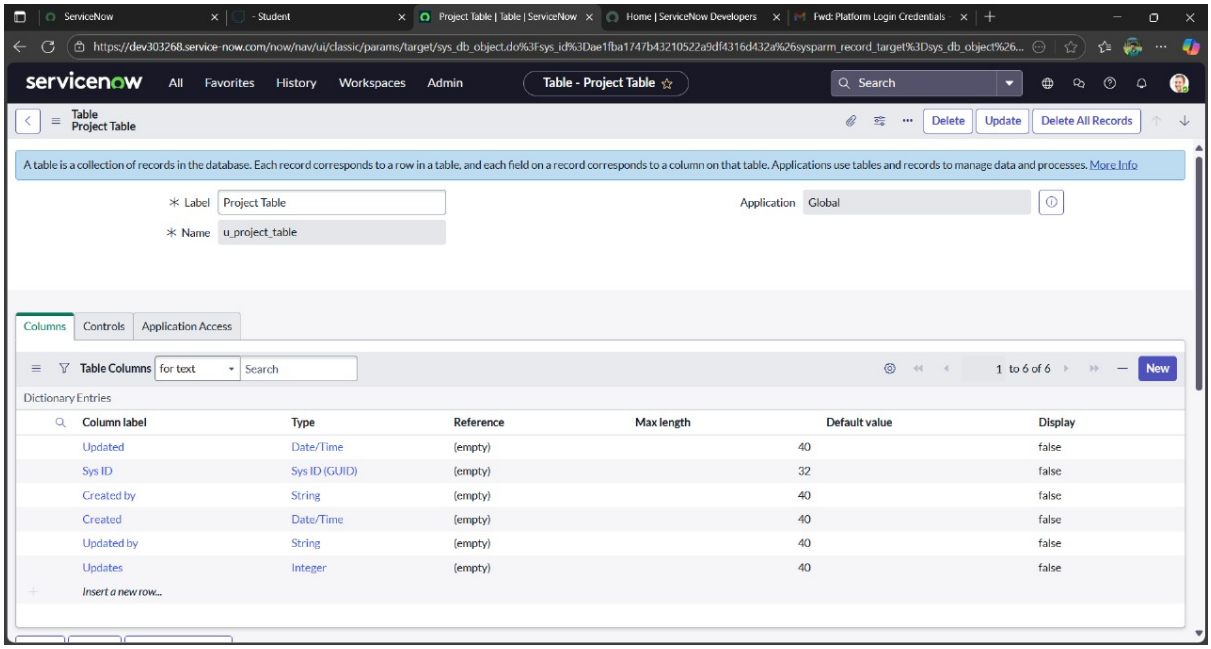

# Role Assignment

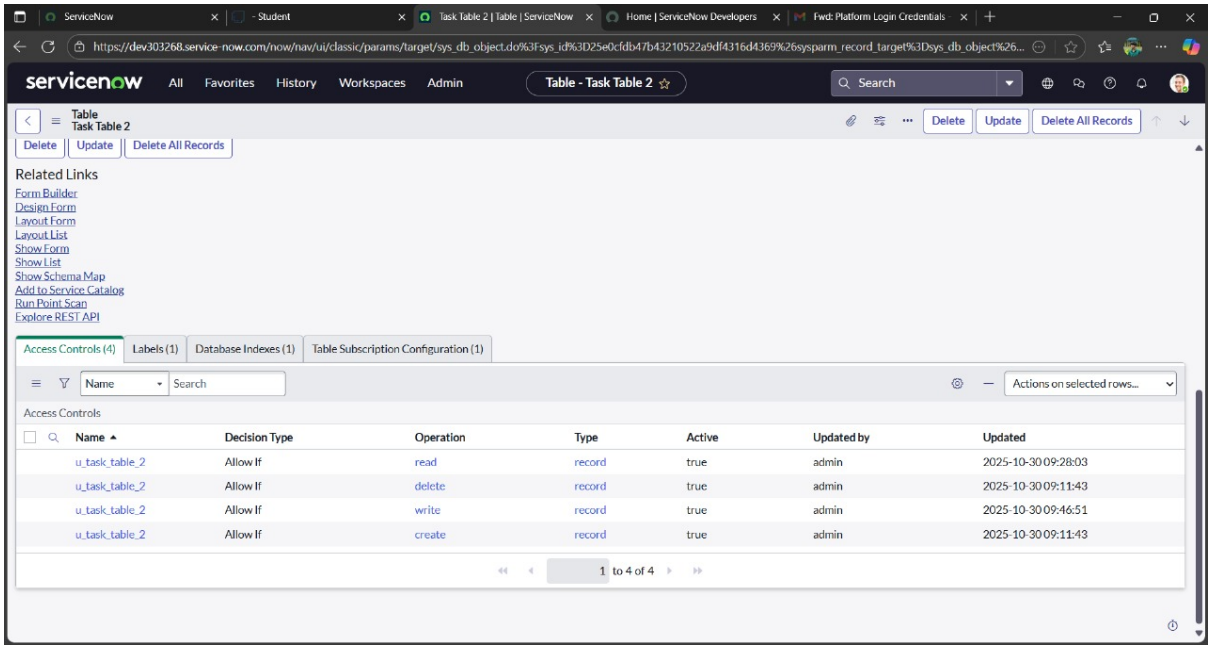



ServiceNow
- Student
Project Member | Role | ServiceNow
Home | ServiceNow Developers
Fwd: Platform Login Credentials
https://dev303268.service-now.com/now/nav/ui/classic/params/target/sys_user_role.do%3Fsys_id%3D5b06be9f47743210522a9df4316d43f2%26sysparm_record_target%3Dsys_user_role%26sy...
servicenow
All
Favorites
History
Workspaces
Admin
Role - Project Member
Search
Role
Project Member
Update
Delete
Name
Project Member
Application
Global
Elevated privilege
Description
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

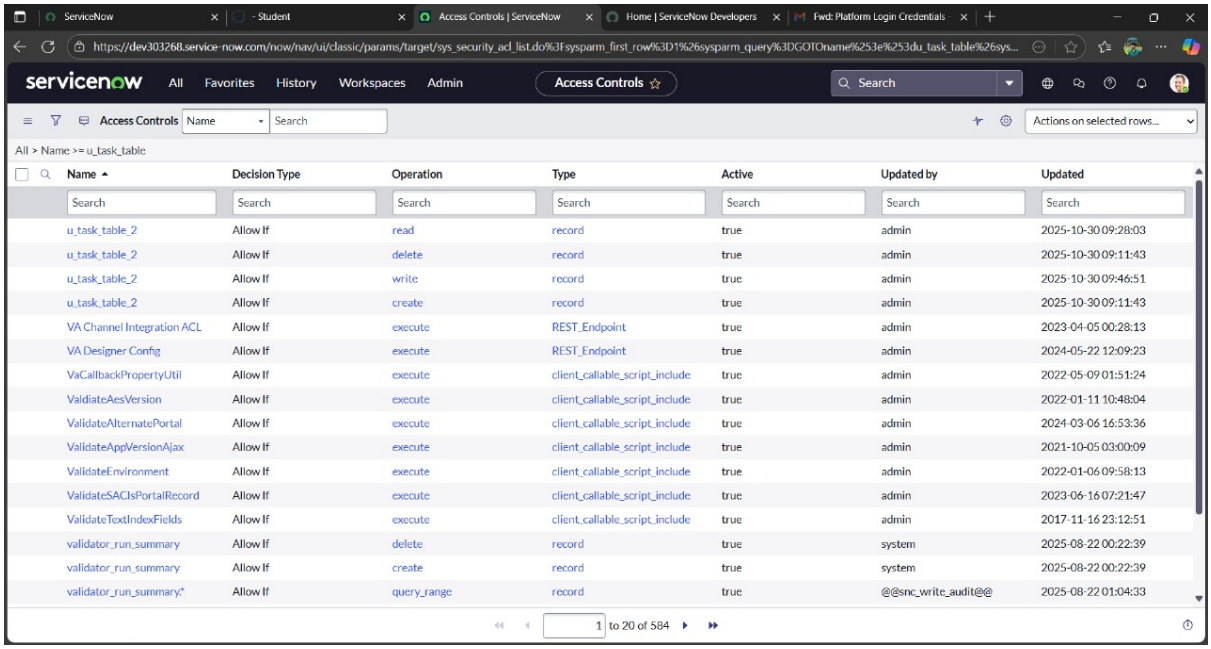

ServiceNow
- Student
team_member | Role | ServiceNow
Home | ServiceNow Developers
Fwd: Platform Login Credentials
https://dev303268.service-now.com/now/nav/ui/classic/params/target/sys_user_role.do%3Fsys_id%3Dd78bf65747b43210522a9df4316d43f8%26sysparm_record_target%3Dsys_user_role%26s...
servicenow
All
Favorites
History
Workspaces
Admin
Role - team_member
Search
Role
team_member
Update
Delete
Name
team_member
Application
Global
Elevated privilege
Description
A basic role for team members who perform assigned tasks under a project.
Update
Delete
Related Links
Run Point Scan
Contains Roles
Applications with Role (1)
Modules with Role
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



## ACL Configuration




**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)