HRIS System Tasks – SharePoint, Power Platform, .NET

Generated on: March 30, 2025

✅ System Task – Geo Spread: Time Zone Handling

The solution architect has to set up consistent time zone handling logic in the .NET backend, Power Automate workflows, and Power Apps interface to standardize time capture and presentation across geographies. This is required to ensure that HR events such as leave requests, approvals, interviews, and onboarding steps are recorded and displayed accurately irrespective of the user's local time zone. During the workflow design and integration phase, the architect must use DateTime.UtcNow in all .NET-based API calls and data persistence layers to ensure that system timestamps are always stored in UTC. In Power Automate flows, the developer must use the convertTimeZone() function to dynamically convert UTC timestamps into each user’s local time zone for email notifications, approvals, or task due dates. In Power Apps, user profile metadata such as time zone must be referenced from either Azure AD or SharePoint user profiles and used to format date-time fields before displaying them on the UI. All time-sensitive data must be stored in UTC format within Dataverse or SQL Server to maintain consistency, and only converted at the presentation layer. This approach helps eliminate workflow misfires, incorrect deadline calculations, and SLA violations that typically occur when timestamps are interpreted in different time zones without normalization. The time zone handling logic must be planned and tested early during the workflow and data model design phase to avoid inconsistencies in later modules such as attendance, leave tracking, and performance evaluations.

✅ System Task – Role-Based Access Control (RBAC)

The solution architect and security lead have to implement granular role-based access control (RBAC) in SharePoint, Power Apps, and the .NET service layer to ensure that each user only accesses the data, pages, workflows, and records relevant to their role. This is required to protect sensitive HR data such as compensation details, performance records, and exit information from unauthorized users while enabling seamless access for employees, managers, HR staff, and administrators based on their responsibilities.

During the security and authorization design phase, the architect must define a clear RBAC matrix that maps user roles (such as Employee, Manager, HR Executive, HR Admin, Finance, IT Admin, Super Admin) to specific permissions (e.g., View Only, Edit Own Records, Approve, Full Admin Rights). In SharePoint, permission levels must be assigned at the site, list, and item level using SharePoint groups or Microsoft 365 security groups. In Power Apps, app-level role-based rules should be enforced using User().Email, Office365Users.MyProfile().JobTitle, and conditional logic within visible/display/editable properties to show or hide content dynamically. In Power Automate, approval flows should branch based on role, and sensitive data must only be routed to authorized users. The .NET backend must implement middleware to check user roles on API requests, using claims-based identity and token-based authorization, typically via Azure AD or OAuth tokens.

This approach ensures regulatory compliance, preserves data confidentiality, and creates a secure, trust-driven environment where users interact only with relevant and authorized components of the HRIS. RBAC must be fully defined and implemented during the system configuration and workflow development phase to avoid rework, data breaches, or access violations in production.

✅ System Task – Department Segregation and Workflow Customization

The business analyst and solution architect have to define the organizational department structure and implement logic in SharePoint, Power Apps, and Power Automate to segregate data, personalize workflows, and enable department-specific operations within the HRIS. This is essential to ensure that users from different departments, such as Finance, IT, Sales, and HR, view only their relevant data, follow workflows tailored to their functions, and operate within access boundaries appropriate to their roles.

During the early design and requirements gathering phase, departments must be captured as metadata fields in all core data entities such as employee profiles, requests, and forms. This metadata should then be used in Power Apps to filter gallery views and limit visibility of records using conditional expressions tied to the user’s department. In Power Automate, conditional branching logic must be applied based on department values to route workflows through department-specific approval chains, notifications, or document templates. SharePoint lists and libraries should be structured with folders or views filtered by department metadata to simplify document access and content management. The .NET backend should enforce department-based filtering in API queries using the authenticated user's claims or roles.

This setup ensures that departments can operate semi-independently within the same system, improves performance by reducing data noise, and enhances user experience by showing only relevant information. It also supports the design of modular workflows, which are easier to maintain and scale. Department segregation and workflow logic must be implemented during the workflow and UI design phase to avoid reengineering when department-level customizations emerge post-launch.

✅ System Task – Organizational Hierarchy and Approval Routing

The business analyst, HR lead, and solution architect have to define the complete organizational hierarchy and integrate it with workflow routing logic in Power Automate, user profile lookups in Power Apps, and access validations in .NET APIs to support manager-based approvals and escalations. This is critical for enabling automated decision flows such as leave approvals, performance reviews, onboarding checklists, and resignation processing, where each task must flow through the appropriate chain of command.

During the planning and configuration phase, the full org structure must be captured either from Active Directory, Azure AD, or uploaded as a reference list into SharePoint or Dataverse. Each employee record must include a direct manager ID or email, and this relationship should be queried dynamically in Power Automate using the Office 365 Users connector to resolve current manager details at runtime. In Power Apps, the manager information should be displayed contextually and used to control form routing and approval buttons. In SharePoint-based scenarios, workflows should resolve manager hierarchy using lookup fields or calculated columns linked to an authoritative source. For .NET-based services, hierarchy metadata must be embedded into JWT tokens or passed as API parameters to ensure that server-side validations respect reporting structures.

This setup ensures that all tasks are routed appropriately without hardcoded assignments, prevents misdirected approvals, and supports escalations to alternate reviewers or higher authorities when managers are unavailable. It also allows HR teams to dynamically adjust workflows when organizational changes occur. Approval routing logic tied to hierarchy must be implemented during the workflow design and employee profile modeling phase to ensure routing logic is scalable, traceable, and audit-compliant across all HR modules.