Software Requirements Specification SRS

Todo List Application with Role-Based Authentication

Version: 1.0
Date: January 23, 2026
Prepared by: Senior Software Architect

Table of Contents

1. Introduction

1.1 Purpose

The primary purpose of this Software Requirements Specification SRS) is to provide a comprehensive and unambiguous definition of the functional and non-functional requirements for the Todo List Application with role-based authentication. This web-based application enables users to securely register, authenticate, and manage personal tasks through create, read, update, and delete CRUD) operations, while supporting role-specific access controls for different user classes.

This document ensures that all stakeholders, including developers, testers, project managers, and endusers, have a clear understanding of the system's expected behavior. By adhering to IEEE Std 830 1998 standards, it facilitates verification and validation processes, minimizes ambiguities, and serves as a contractual reference for system development. The SRS emphasizes secure task organization and persistence, allowing users to filter tasks by status and maintain productivity without data loss.

Furthermore, this specification delineates the boundaries of the system, focusing solely on the defined features to prevent scope creep. It supports the development of a reliable, user-friendly application that meets organizational needs for task management in a multi-user environment.

1.2 Scope {#scope}

The Todo List Application is a web-based system designed to support user registration, login with rolebased authentication (admin, super admin, user), and task management operations including creation, editing, deletion, marking as complete/incomplete, filtering by status (all, active, completed), and clearing completed tasks. All task data persists in a MySQL database and is visible only to the authenticated user or authorized roles.

The scope explicitly excludes advanced features such as team collaboration, real-time notifications, mobile native applications, integration with external calendars, or advanced reporting. The system provides responsive design for desktop and mobile web browsers but does not include dedicated mobile apps. Role-specific dashboards ensure that users see personal tasks, admins view all tasks, and super admins have full system oversight including user management.

This application differentiates itself through its emphasis on secure, persistent storage and intuitive rolebased interfaces, making it suitable for individual and administrative task tracking

within small to medium-sized teams.

1.3 Definitions, Acronyms, and Abbreviations {#definitions}

- **CRUD**: Create, Read, Update, Delete – Fundamental operations for data manipulation. **Dashboard**:
- Role-specific web page displaying tasks or management interfaces post-login.
- **Role-Based Access Control RBAC** : Mechanism restricting system access based on user roles User, Admin, Super Admin).

- **Active Tasks**: Tasks not marked as complete.
- **Completed Tasks**: Tasks marked as complete by the user.
- **Passport.js**: Node.js middleware for authentication.
- **MySQL**: Relational database management system for data persistence. **SRS**:
- Software Requirements Specification.
- **IEEE**: Institute of Electrical and Electronics Engineers. **UI**:
- User Interface.

1.4 References {#references}

- IEEE Std 830 1998, Recommended Practice for Software Requirements Specifications.

GitHub Repository: https://github.com/Onkar-Birajdar/Todo-List

- Studocu IEEE SRS Format: https://www.studocu.com/in/document/niit-university/computer-s cience/the-ieee-srs-format-overview-and-key-sections-explained/128695407
- Scribd SRS IEEE Format: https://www.scribd.com/document/823700295/SRS IEEEFormat

Additional references as listed in the original specification.

1.5 Document Overview {#document-overview}

This SRS is organized into seven main sections. Section 1 introduces the document's purpose and structure. Section 2 provides an overall description of the product. Section 3 details functional requirements with precise inputs, behaviors, and outputs. Section 4 covers non- functional requirements. Sections 5 7 explain supporting diagrams, including use case, workflow, and data flow diagrams. Each section uses consistent numbering and formal language to ensure clarity and traceability for PDF export and stakeholder review.

2. Overall Description {#overall-description}

2.1 Product Perspective {#product-perspective}

The Todo List Application builds upon traditional task management tools by incorporating robust multirole authentication using Passport.js and role-specific dashboards. Unlike basic todo lists that lack persistence or security, this system ensures tasks are securely stored in MySQL,

accessible only to authorized users. It positions itself as a secure, scalable web application for personal and administrative task oversight.

From a market perspective, it addresses gaps in open-source todo apps by providing granular role controls: standard users manage personal tasks, admins oversee all tasks, and super admins handle user lifecycle. This enhances productivity in environments requiring accountability without introducing unnecessary complexity. The application evolves standard task managers into enterprise-ready tools while maintaining simplicity.

2.2 Product Functions {#product-functions}

The core functions of the Todo List Application are strictly limited to those explicitly defined:

- User signup with username, password, and role selection.
- Authentication and login redirecting to role-specific dashboards.
- Task CRUD operations: create, edit, delete, and mark as complete/incomplete. Task filtering by status (all, active, completed) and clearing completed tasks.
- Persistent storage of tasks in MySQL, visible only to the logged-in user or authorized roles.

These functions enable seamless task lifecycle management, from creation to archival, within a secure, authenticated environment. Each function integrates with role-based views to ensure appropriate access levels.

2.3 User Classes and Characteristics {#user-classes}

The system defines three distinct user classes, each with specific characteristics and access levels:

- **User**: Basic end-users with low to moderate technical expertise. They perform personal task management CRUD, filter, clear) via intuitive web interfaces. Access limited to own tasks; typical users are non-technical individuals like managers or students using desktop or mobile browsers.
- **Admin**: Moderately technical users responsible for oversight. They access all users' tasks, perform task management, and view comprehensive dashboards. Suitable for team leads monitoring productivity.
- **Super Admin**: Highly technical users with full system control. They manage users (implied through rolespecific branches), access all dashboards, and oversee the entire system. Intended for system administrators.

All users interact via standard web browsers, assuming basic familiarity with web navigation.

2.4 Operating Environment {#operating-environment}

The application operates in a client-server architecture:

- **Client Side**: HTML, CSS, JavaScript, and Bootstrap for responsive UI, compatible with modern web browsers on desktops and mobiles.
- **Server Side**: Node.js with Express.js framework handling backend logic.

Database: MySQL for persistent storage of users and tasks.

- **Deployment**: Cloud platforms like Heroku, requiring stable internet connectivity.

The environment supports concurrent access without platform-specific dependencies beyond web standards.

2.5 Design and Implementation Constraints {#constraints}

- Mandatory technology stack: Node.js, Express.js, MySQL, Passport.js for authentication.

Responsive design using Bootstrap; no custom frameworks.

- No external APIs or third-party services beyond the specified database.

Web-only deployment; no native apps or desktop versions.

- Compliance with IEEE SRS standards for documentation.

These constraints ensure consistency, security, and ease of maintenance.

2.6 Assumptions and Dependencies {#assumptions}

Assumptions:

- Users have stable internet access and modern browsers supporting JavaScript.

MySQL server is available and properly configured.

- End-users possess basic web literacy for navigation.

Dependencies:

- NPM packages: Express, Passport.js, MySQL driver. Browser
- compatibility with Chrome, Firefox, and equivalents.

3. Functional Requirements {#functional-requirements}

Each functional requirement is elaborated with detailed description, inputs, system behavior, and outputs, using traceable numbering FR 1 through FR 5 .

3.1 FR 1 User Signup {#fr-1}

Description: The system shall allow new users to register by providing a username, password, and selecting a role User, Admin, Super Admin). This establishes a persistent user account in the MySQL database.

Inputs:

- Username (unique string).
- Password (hashed string).
- Role selection (dropdown: User, Admin, Super Admin).

System Behavior:

- Validates username uniqueness via MySQL query.

Hashes password using Passport.js.

- Inserts user record into Users data store if valid.

Handles duplicates by displaying error message.

Redirects to login page upon success.

Outputs:

- Success message: "Account created successfully."
- Error message for duplicates: "Username already exists." New session-
- ready user profile.

3.2 FR 2 User Authentication and Login {#fr-2}

Description: The system shall authenticate users upon login and redirect them to a role-specific dashboard.

Inputs:

- Username.

Password.

System Behavior:

- Queries MySQL Users store for matching credentials.

Validates password hash via Passport.js.

- Establishes session with role assignment.
- Redirects: User to personal dashboard, Admin to all-tasks view, Super Admin to full management dashboard.
- Invalid credentials trigger error display and retry option. **Outputs**:
- Role-specific dashboard HTML page. Error
- message: "Invalid credentials."

3.3 FR 3 Task Management Operations {#fr-3}

Description: Authenticated users shall create, edit, delete tasks, and mark them as complete or incomplete, with visibility restricted by role.

Inputs:

- Task details: title, description (for create/edit). Task
- ID (for edit/delete/mark).

- Completion status toggle.

System Behavior:

- For create: Inserts into MySQL Tasks store, associated with user ID. For edit/delete/mark:

Updates/deletes specific record via Task ID.

- Role enforcement: Users access own tasks; Admin/Super Admin access all. Realtime

UI updates post-operation.

Outputs:

- Updated task list on dashboard.
- Confirmation messages: "Task created/updated/deleted/marked."

3.4 FR 4 Task Filtering and Clearing {#fr-4}

Description: The system shall filter tasks by status (all, active, completed) and allow clearing of completed tasks.

Inputs:

- Filter selector: "All", "Active", "Completed".

Clear request (button for completed tasks).

System Behavior:

- Queries MySQL Tasks store with status filter.

  Displays filtered list dynamically.

- On clear: Deletes all completed tasks for the user/role scope.

  Refreshes dashboard post-clear.

Outputs:

- Filtered task list view.
- Confirmation: "Completed tasks cleared."

3.5 FR 5 Data Persistence {#fr-5}

Description: Tasks and user data shall persist in MySQL and be visible only to the logged-in user.

Inputs:

- Any task/user operation.

System Behavior:

- All CRUD operations commit to MySQL.
- Queries scoped by user ID/role.
- Ensures data integrity via transactions.

Outputs:

- Persistent, role-restricted data views.

4. Non-Functional Requirements

4.1 Performance

The system shall handle up to 100 concurrent users with task operations CRUD, filter) responding in under 1 second. Database queries must optimize for low latency, achieving 99th percentile response times below 500ms under load. Page load times on dashboards shall not exceed 2 seconds for typical use.

4.2 Security

Passwords shall be hashed using Passport.js before storage. Role-based access control enforces least privilege: Users see only personal tasks; higher roles extend visibility. Sessions must timeout after inactivity, with HTTPS enforcement in production. No unauthenticated access to dashboards or data.

4.3 Usability

The UI shall be responsive across mobile and desktop using Bootstrap, with intuitive elements like hover effects for edit/delete. Navigation must be straightforward, with clear feedback for all actions (e.g., success/error modals). Accessibility features include keyboard navigation and high-contrast modes where standard.

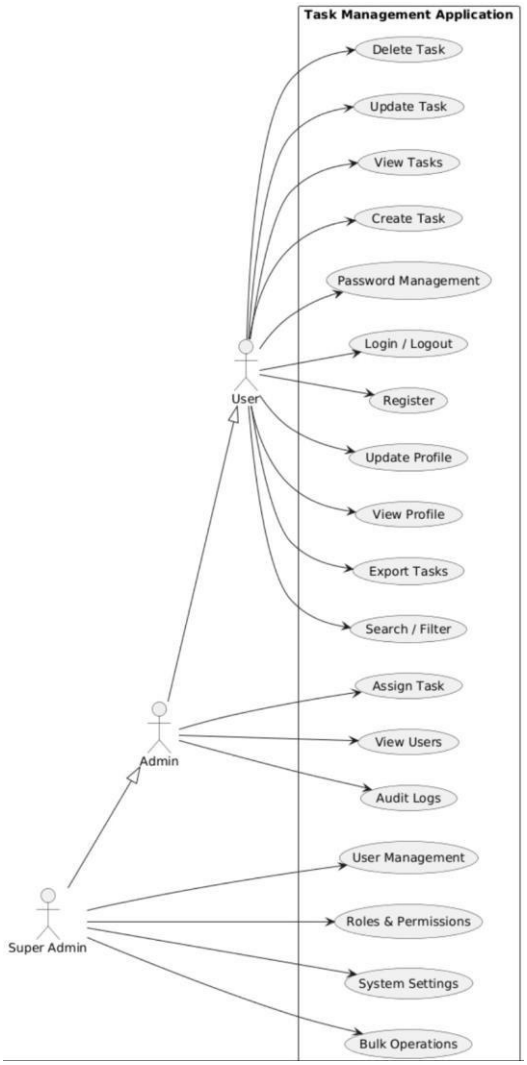
4.4 Reliability

The system shall maintain 99% uptime, with automatic data backups via MySQL. Error handling ensures graceful degradation; no single failure causes total outage. Data consistency preserved through ACIDcompliant transactions.

4.5 Portability

The web application shall function across major browsers Chrome, Firefox) without OS dependencies. Responsive design ensures compatibility with devices from 320px to 4K resolutions. Deploymentagnostic for Heroku-like platforms.

5. Use Case Diagram Explanation



System: Task Management Application

Actors Identified

The use case diagram represents three actors with a clear hierarchical relationship:

1. **User** – Primary actor
2. **Admin** – Specialized actor with extended privileges over User
3. **Super Admin** – Highest-privileged actor with system-level control

The hierarchy indicates that **Admin inherits User capabilities**, and **Super Admin inherits Admin capabilities**, in addition to having exclusive system functions.

User Actor – Use Cases

The **User** actor interacts with the Task Management Application to perform personal task and accountrelated operations. The following use cases are directly associated with the User:

- **Register**
  Allows a new user to create an account in the system.
- **Login / Logout**
  Enables the user to authenticate into the system and terminate the session when required.
- **Password Management**
  Allows the user to manage password-related actions as shown in the diagram.
- **View Profile**
  Enables the user to view personal profile information.
- **Update Profile**
  Allows modification of user profile details.
- **Create Task**
  Enables the user to create a new task.
- **View Tasks**
  Allows the user to view the list of existing tasks.
- **Update Task**
  Enables modification of existing tasks.
- **Delete Task**
  Allows removal of tasks from the system.
- **Search / Filter**
  Enables searching and filtering of tasks based on criteria.
- **Export Tasks**
  Allows the user to export task data.

These use cases collectively define the **core task management and account management capabilities** available to a standard user.

Admin Actor – Use Cases

The **Admin** actor extends the User role and has access to administrative functions related to task oversight and user visibility. In addition to inherited User use cases, the Admin is directly associated with:

- **Assign Task**
  Allows the admin to assign tasks, as depicted in the diagram.
- **View Users**
  Enables the admin to view user information within the system.
- **Audit Logs**
  Allows access to audit logs for monitoring and review purposes.

These use cases indicate that the Admin role focuses on **supervision, monitoring, and task coordination**.

Super Admin Actor – Use Cases

The **Super Admin** actor represents the highest authority within the system. This actor is associated with system-level management use cases, including:

- **User Management**
  Enables management of user accounts.

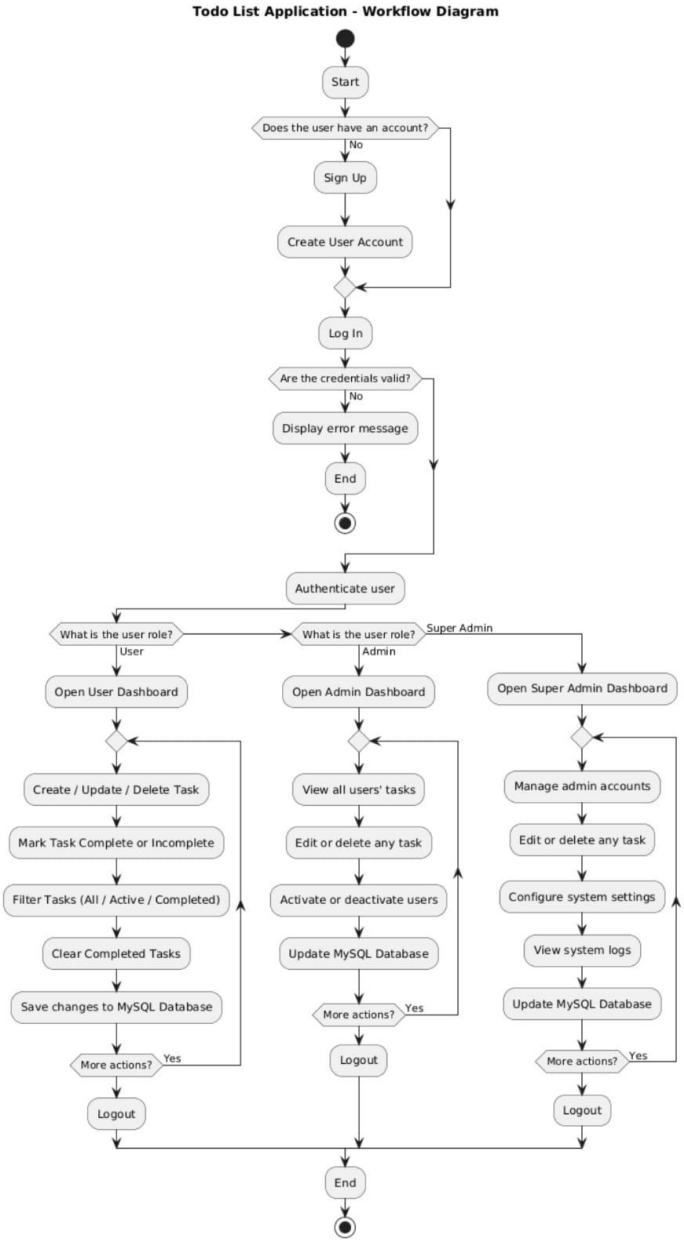
- **Roles & Permissions**
  Allows configuration and management of roles and access permissions.
- **System Settings**
  Enables modification of system-level configuration settings.
- **Bulk Operations**
  Allows execution of bulk actions as shown in the diagram.

These use cases indicate that the Super Admin is responsible for **system configuration, access control, and administrative governance**.

System Boundary

All use cases are enclosed within the **Task Management Application** system boundary, indicating that each action is handled internally by the system.

Workflow Diagram Explanation



System: Task / Todo Management Application

The workflow diagram illustrates the **end-to-end operational flow** of the system, covering **authentication, task operations, administrative actions, and data persistence**. The diagram is organized into **logical workflow zones**, each representing a major functional stage of the system.

1. User Authentication & Account Setup Flow

The workflow begins when the user accesses the Todo List Application.

1. The system starts the application workflow.
2. The system checks whether the user already has an account.
3. If the user does not have an account:

- The user is directed to the **Sign Up** process.
- A new user account is created and stored in the system.

4. After account creation (or if the user already has an account), the user proceeds to **Log In**.
5. The system validates the entered login credentials.
6. If the credentials are invalid:

- An error message is displayed.
- The workflow terminates.

7. If the credentials are valid:

- The user is successfully authenticated.
- The system proceeds to role identification.

This stage ensures that only authenticated users can access the application features.

---

2. Role Identification and Access Routing

Once authentication is successful, the system determines the user's role.

1. The system identifies the authenticated user's role as one of the following:

- User
- Admin
- Super Admin

2. Based on the identified role:

- The user is redirected to the corresponding dashboard.
- Only role-specific actions are made available.

This decision point controls access and enforces role-based permissions throughout the system.

---

3. Task Management Workflow (User Operations)

When the user role is identified as **User**, the following workflow is executed:

1. The **User Dashboard** is opened.
2. The user can perform task-related operations:

- Create tasks
- Update existing tasks
- Delete tasks

3. The user can:

- Mark tasks as **Complete** or **Incomplete**
- Filter tasks (All / Active / Completed)
- Clear completed tasks

4. All task changes are saved to the **MySQL database**.
5. After each action:

- The system checks if the user wants to perform more actions.
- If yes, the workflow loops back to the dashboard.
- If no, the user logs out and the workflow ends.

This workflow enables users to fully manage their personal tasks.

4. Administrative Workflow (Admin Actions)

When the authenticated role is **Admin**, administrative functionality becomes available.

1. The **Admin Dashboard** is opened.
2. The admin can:

- View all users' tasks
- Edit or delete any task in the system
- Activate or deactivate user accounts

3. All administrative changes are saved to the **MySQL database**.
4. The system checks if more actions are required:

- If yes, the admin continues managing data.
- If no, the admin logs out and exits the system.

This workflow allows administrators to oversee users and maintain task integrity.

---

5. System Management Workflow (Super Admin Actions)

When the role is identified as **Super Admin**, system-level controls are enabled.

1. The **Super Admin Dashboard** is opened.
2. The super admin can perform advanced operations:

- Manage admin accounts
- Edit or delete any task
- Configure system settings
- View system logs

3. All configuration and management updates are stored in the **MySQL database**.
4. The system checks for additional actions:

- If yes, the workflow continues.
- If no, the super admin logs out and the workflow ends.

These actions are restricted to ensure system security and stability.

---

6. Data Interaction and Persistence Flow

Throughout all workflow paths:

- User data, task data, and system data are:
- Retrieved from the MySQL database
- Modified based on user actions
- Saved back to the database
- Each operation follows a controlled sequence:
- Action request → Validation → Database update → Confirmation

This guarantees data consistency and reliable system behavior.

---

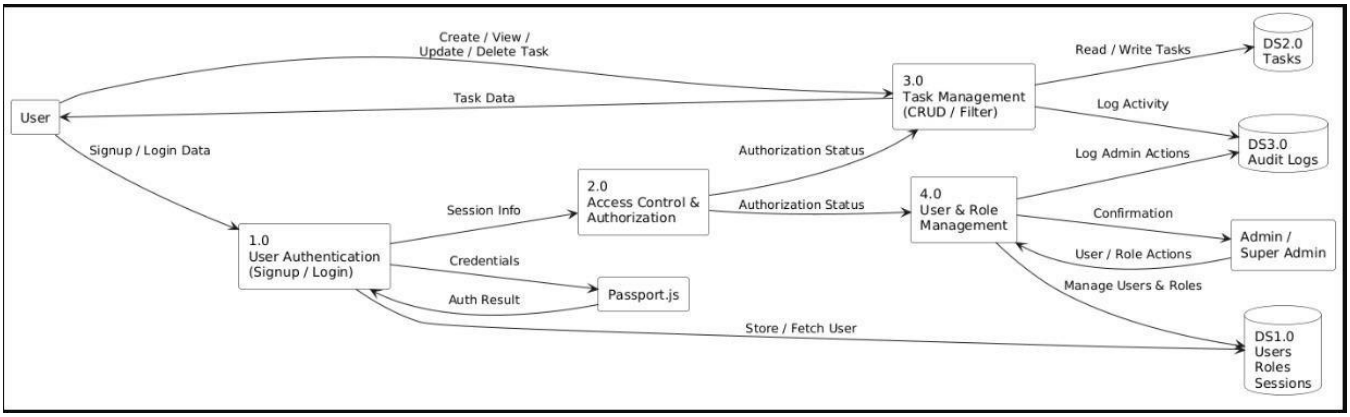
7. Workflow Completion and Logout

The workflow concludes when:

- The user, admin, or super admin chooses to log out.
- The system terminates the active session.
- The application reaches the **End** state.

This ensures secure session handling and proper termination of access.

6 . Data Flow Diagram DFD Explanation



System: Task Management Application **Diagram**
Level: Level-1 DFD

1. **External Entity: User**

The **User** is the primary external entity interacting with the system.

Data Provided by User

- **Signup / Login Data**
- **Task Data** (Create, View, Update, Delete actions)

Data Received by User

- **Authorization Status**
- **Task Operation Results**

2. **Process 1.0 – User Authentication (Signup / Login)**

This process handles user authentication.

Input Data

- Signup / Login Data (from User)

Internal Flow

- Credentials are sent to **Passport.js** for authentication.
- Authentication result is returned.

Output Data

- **Session Info** → sent to Process 2.0
- **Auth Result** → sent back to the User
- **Store / Fetch User** → interacts with DS1.0 (Users, Roles, Sessions)

3. External Component – Passport.js

Passport.js acts as the **authentication mechanism**.

Input

- User credentials

Output

- Authentication result (success/failure) returned to Process 1.0

4. Process 2.0 – Access Control & Authorization

This process validates **user permissions** after authentication.

Input Data

- Session Info (from Process 1.0)

Output Data → Authorization Status →

sent to:

  - Process 3.0 (Task Management)
  - Process 4.0 (User & Role Management)

This ensures that **only authorized users can access further operations**.

5. Process 3.0 – Task Management (CRUD / Filter)

This process handles all task-related operations.

Input Data

- Task Data (from User)
- Authorization Status (from Process 2.0)

Operations

- Create Task
- View Task
- Update Task
- Delete Task
- Filter Tasks

Output Data

- Read / Write Tasks → DS2.0 (Tasks)
- Log Activity → DS3.0 (Audit Logs)
- Task results → returned to User

6. Process 4.0 – User & Role Management

This process manages users and roles.

Input Data

- Authorization Status (from Process 2.0)
- User / Role Actions (from Admin / Super Admin)

Operations

- Manage Users
- Manage Roles

Output Data

- Confirmation → Admin / Super Admin
- Manage Users & Roles → DS1.0 (Users, Roles, Sessions)
- Log Admin Actions → DS3.0 (Audit Logs)

7. Data Stores

DS1.0 – Users, Roles, Sessions

- Stores user accounts
- Role assignments
- Session information

DS2.0 – Tasks

- Stores task records
- Supports read/write operations

DS3.0 – Audit Logs

- Stores:
    - Task activity logs
    - Admin action logs

8. External Entities: Admin / Super Admin

Admin and Super Admin interact with **User & Role Management**.

Actions

- Perform user management
- Manage roles

Output

Receive confirmation for management actions