

Community App

User Access

**Contents**

[1 Overview 3](#_Toc205216782)

[1.1 Overview 3](#_Toc205216783)

[1.2 Permission Roles 3](#_Toc205216784)

[2 Scenarios 3](#_Toc205216785)

[3 Business Requirements 5](#_Toc205216786)

[4 Functional Requirements 5](#_Toc205216787)

[5 Dependencies 6](#_Toc205216788)

[6 Checklist 6](#_Toc205216789)

[7 Change and Review history 6](#_Toc205216790)

**Quick Spec – User Access**

Author: Tahta Alfionita (tahta@slerate.com)

This document is intended to be used as part of the SLERATE spec Development and Review Process.

Confidentiality Notice

This document is an internal working document of SLERATE. It contains trade secrets and other proprietary information, which are the confidential property of SLERATE.  Neither this document nor the information contained within is to be distributed—in whole or part— outside of the SLERATE team by any means without prior authorization, including but not limited to printed, magnetic, electronic or verbal forms.

1. Overview

## Overview

The User Access feature in the Admin Panel defines the authentication, authorization, and user permissions mechanism (for admins, managers, staff) to ensure only authorized users can access and operate valid modules and features in the admin panel. The purpose is to secure data and enforce proper access control aligned with user roles and responsibilities.

## Permission Roles

1. Manager: Full access through admin panel
2. Admin: Member database management
3. Content creator: Content management
4. Scenarios

|  |  |
| --- | --- |
| **Scenario Name** | **User Login** |
| Background | Manjit wants to login to admin panel as a community manager to see if there is any update in there |
| Objective | Provide a secure, reliable login mechanism with clear feedback and protection against unauthorized access attempts. |
| Persona(s) | Manjit – Manager |
| Scenario Steps | 1. Manjit opens the Admin Panel login page. 2. Manjit inputs username/email and password. 3. System validates credentials against the user database. 4. If valid, system creates a user session and redirects the user to their role-based dashboard. 5. If invalid, system increments the failed login attempt counter. 6. If failed login attempts reach 5 in a row, system locks the account for 15 minutes and notifies the user. 7. Error Handling:    1. Invalid credentials trigger error messages like “Invalid username or password.”    2. Locked account displays: “Account temporarily locked due to too many failed attempts. Please try again in 15 minutes.”    3. System prevents login during lockout period. |

|  |  |
| --- | --- |
| **Scenario Name** | **Account Locked Due to Excessive Failed Login Attempts** |
| Background | Manjit wants login to his account but forgot his password, he tried to log in using the last password he remembered, but after five attempts, his account got locked. |
| Objective | To prevent brute-force attacks and unauthorized access attempts, the system temporarily locks accounts after repeated failed login attempts. |
| Persona(s) | Manjit – Manager |
| Scenario Steps | 1. Manjit tries to log in after 5 consecutive failed attempts. 2. System denies login and displays a lockout message specifying the duration (15 minutes). 3. Manjit must wait for the lockout period to expire before retrying. 4. Error Handling:    1. Attempts to log in during the lockout period receive immediate denial and explanatory messages.    2. System logs lockout events. |

|  |  |
| --- | --- |
| **Scenario Name** | **Role-Based Dashboard Access** |
| Background | Every user wants to login to admin panel and dashboard for every role is different |
| Objective | Direct users to their specific dashboards post-login and restrict access to other role areas to maintain security and usability. |
| Persona(s) | Manjit – Manager  Adam – Admin  Connie – Content creator |
| Scenario Steps | 1. User logs in successfully. 2. System identifies the user role from profile. 3. System redirects to the dashboard matching the user role:    1. Administrator → Administrator Dashboard    2. Manager → Manager Dashboard    3. Content Creator → Content Creator Dashboard 4. If user attempts to access a different role’s dashboard URL, system denies access and displays “Access Denied.” 5. Error Handling:    1. Unauthorized access attempts are blocked and logged.    2. Provide user-friendly “Access Denied” messages. |

1. Business Requirements

|  |  |  |
| --- | --- | --- |
|  | **Business Requirement** | **Priority** |
| 1 | The system must provide a login page for Admin Panel user authentication. | High |
| 2 | The system must lock the user account temporarily after 5 consecutive failed login attempts for 15 minutes. | High |
| 3 | Locked accounts must display a clear message: “Account temporarily locked due to too many failed attempts. Please try again in 15 minutes.” | High |
| 4 | Access to dashboards must be role-based: Administrator, Manager, Content Creator. | High |
| 5 | Each dashboard must display features and information appropriate to the user role. | High |
| 6 | Access security must prevent unauthorized access and maintain data integrity. | High |

1. Functional Requirements

|  |  |  |
| --- | --- | --- |
|  | **Functional Requirement** | **Business Requirement(s)** |
| 1 | The login page must allow users to enter username and password. | 1 |
| 2 | After 5 consecutive failed login attempts, the system must lock the account for 15 minutes and display a lockout message. | 2, 3 |
| 3 | The system must prevent login attempts on locked accounts until the lock period expires. | 3 |
| 4 | Upon successful login, the system must redirect users to their respective dashboards: Administrator Dashboard, Manager Dashboard, or Content Creator Dashboard. | 4, 5 |
| 5 | The system must display clear and informative error messages on authentication failures. | 1, 3 |

1. Dependencies

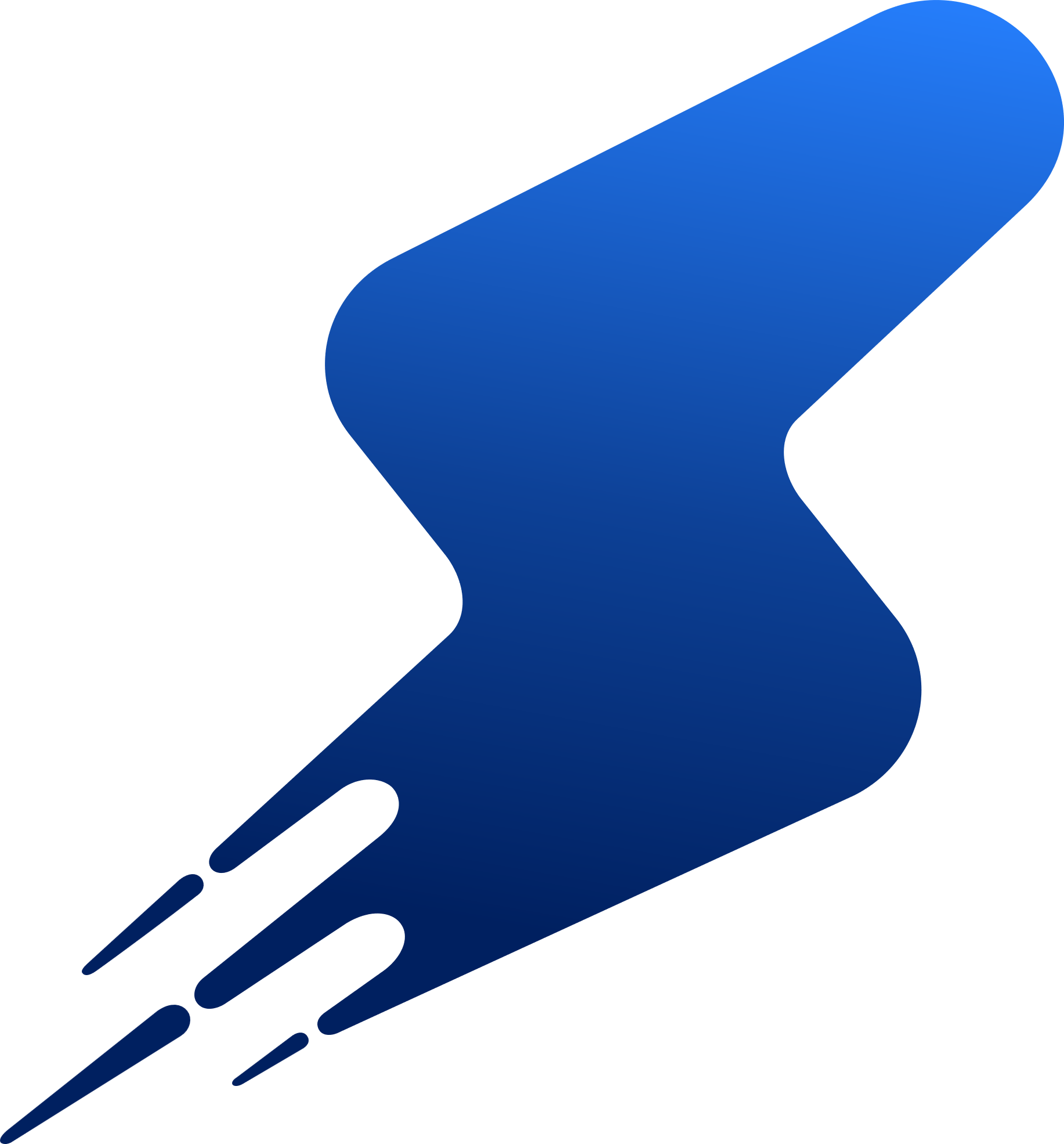
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Team/Contact** | **Type** | **Description** |
| 1 | Backend Developer | Technical | Implements role-based access control (RBAC) logic and permission gates |
| 2 | Frontend Developer | Technical | Hides/disables features in UI based on user role |

1. Checklist

|  |  |  |
| --- | --- | --- |
|  | **Checklist Item** | **Response** |
| 1 | Has this functional spec been reviewed by Dev? | No |
| 2 | Has this functional spec been reviewed by UI? | No |
| 3 | Has this functional spec been reviewed by Manager? | No |
| 4 | All user roles are clearly defined (Admin, Manager, Member, Moderator, etc.) | No |
| 5 | Functional permissions for each role are documented and approved | No |
| 6 | Role-based access control is enforced at both API and UI levels | No |
| 7 | Users can only access features/screens assigned to their role | No |
| 8 | Unauthorized access attempts are blocked and return appropriate error messages | No |
| 9 | New roles or permissions can be added without refactoring core logic | No |
| 10 | User roles are stored securely in the database and loaded on login | No |
| 11 | Manager can view and update user roles from admin panel | No |
| 12 | All access events and permission changes are logged for audit | No |

1. Change and Review history

| **Date** | **Author, editor, or Reviewer** | **Description of change(s)/Comments on review** |
| --- | --- | --- |
| 4/8/2025 | Tahta (Author) | Create this quick specs document |
|  |  |  |
|  |  |  |



SLERATE

August 2025

Copyright © SLERATE

www.SLERATE.com