

Community App

Member Database

**Contents**

[1 Overview 3](#_Toc204849367)

[2 Scenarios 3](#_Toc204849368)

[3 Business Requirements 5](#_Toc204849369)

[4 Functional Requirements 5](#_Toc204849370)

[5 Dependencies 6](#_Toc204849371)

[6 Checklist 6](#_Toc204849372)

[7 Change and Review history 6](#_Toc204849373)

**Quick Spec – Member Database**

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

The Member Database module in the Admin Panel provides administrators with complete management capabilities over the community members’ data. Admins can view, search, filter, create, update, and delete member records, as well as manage member statuses and roles. This module enables efficient oversight and control of the member base, ensuring data integrity, privacy compliance, and user management aligned with community policies.

1. Scenarios

|  |  |
| --- | --- |
| **Scenario Name** | **Admin/Manager Searches Member Database** |
| Background | Admin/manager needs to quickly find specific members using keywords like name or email. |
| Objective | Enable admin/manager to search the member database by various keywords accurately and efficiently. |
| Persona(s) | Adam – Admin  Manjit – Manager |
| Scenario Steps | 1. Adam/Manjit logs into the admin panel. 2. Adam/Manjit navigates to the “Member Database” page. 3. Adam/Manjit sees the list of all members with relevant info. 4. Adam/Manjit types a keyword (member name and email) into the search bar.      1. System validates the input fields: checks for invalid characters or unsupported formats. 2. If invalid input detected, system immediately displays a clear error message, e.g., “Invalid search term format.” and prevents query execution. 3. If input is valid, the system sends query to backend to retrieve data. 4. If backend returns no matching members, system shows “No members found matching your criteria.” message. 5. If backend or network error occurs, system displays a generic error message like “Unable to retrieve member data. Please try again later.” and logs the error internally. 6. Adam/Manjit can retry search/filter after correcting input or when system is restored. 7. If validation is success the system filters and displays only members matching the search keyword. 8. Adam/Manjit reviews the filtered results and selects a member record if needed. 9. Adam/Manjit proceeds with further actions such as viewing or updating member data if required. |

|  |  |
| --- | --- |
| **Scenario Name** | **Admin/Manager Filters Members by Status (Pending, Approved, Rejected)** |
| Background | Admin/manager needs to view members based on their status to manage workflows effectively. |
| Objective | Allow admin/manager to filter the member list by status categories like Pending, Approved, or Rejected to streamline member management. |
| Persona(s) | Adam – Admin  Manjit – Manager |
| Scenario Steps | 1. Adam/Manjit logs into the admin panel. 2. Adam/Manjit navigates to the “Member Database” section. 3. Adam/Manjit locates the status filter option 4. Adam/Manjit selects one status to filter (Pending, Approved, Rejected).      1. The system refreshes the member list showing only members matching the selected status or statuses. 2. Adam/Manjit scans through filtered members and can further select any member to view or manage details. 3. Adam/Manjit may change the filter selection to view other status categories as needed. |

1. Business Requirements

|  |  |  |
| --- | --- | --- |
|  | **Business Requirement** | **Priority** |
| 1 | Admins/manager must be able to view the complete member database with detailed member data. | High |
| 2 | Admins/manager must have search and filter capabilities to locate members quickly. | High |
| 3 | The system must support status management (accepted, rejected, pending). | High |
| 4 | CRUD operations and sensitive actions should be permission-restricted by role. | High |
| 5 | The system should log admin and manager activities for audit purposes. | High |
| 6 | Data privacy policies must be respected, securing sensitive personal information. | High |
| 7 | The interface should be user-friendly and responsive for ease of use across devices. | Medium |

1. Functional Requirements

|  |  |  |
| --- | --- | --- |
|  | **Functional Requirement** | **Business Requirement(s)** |
| 1 | Admin/manager can access a dashboard screen listing all members with pagination, showing key member info (name, email, status). | 1 |
| 2 | Admin/manager can perform keyword search by member name, email, or ID. | 2 |
| 3 | Admin/manager can apply filters such as status (pending, accepted, rejected) | 2 |
| 4 | Admin/manager can click on a member record to view detailed profile information, including registration data and activity logs. | 1 |
| 5 | System enforces role-based access control (RBAC) to restrict admin capabilities based on their assigned roles. | 4 |
| 6 | Admin/manager actions such as update, approving, rejecting are logged with timestamp and admin ID for auditing. | 5 |
| 7 | Sensitive data such as passwords or personal identifiable information (PII) is handled securely and masked as necessary in the UI. | 6 |
| 8 | The member database interface is responsive and accessible on different screen sizes and devices. | 7 |
| 9 | System validates all admin/manager input fields with proper error messages on invalid data. | 3 |

1. Dependencies

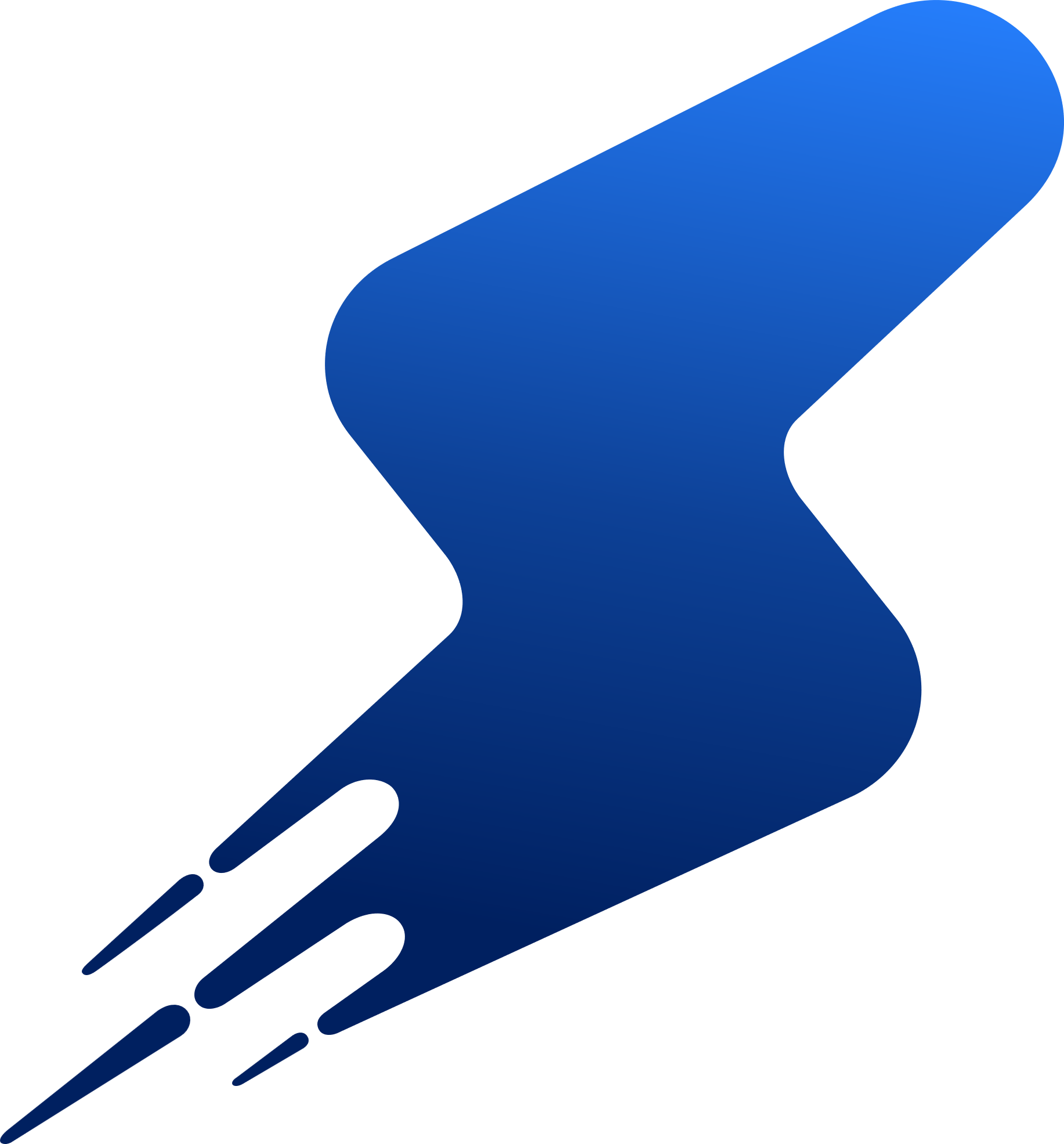
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Team/Contact** | **Type** | **Description** |
| 1 | Backend Developer | Technical | Design, build, and maintain the member database schema and queries |

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 | Database schema for member data is finalized and documented | Yes |
| 5 | Data validation rules (e.g., required fields, data types) applied | Yes |
| 6 | Secure authentication and access control to member database enabled | Yes |
| 7 | Backup and recovery strategy is in place | Yes |
| 8 | API endpoints to access, create, update, and delete member data ready | Yes |

1. Change and Review history

| **Date** | **Author, editor, or Reviewer** | **Description of change(s)/Comments on review** |
| --- | --- | --- |
| 30/7/2025 | Tahta (Author) | Create the quick specs document |
|  |  |  |
|  |  |  |



SLERATE

Month 202x

Copyright © SLERATE

www.SLERATE.com