

Community App

Member Sign Up

**Contents**

[1 Overview 3](#_Toc1821189554)

[2 Scenarios 3](#_Toc1502503580)

[3 Business Requirements 13](#_Toc1734218969)

[4 Functional Requirements 13](#_Toc191564253)

[5 Dependencies 14](#_Toc437123860)

[6 Checklist 14](#_Toc958413644)

[7 Change and Review history 15](#_Toc566035261)

**Quick Spec – Member Sign Up**

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 Sign-Up feature provides a streamlined, secure registration flow for new users to become members of the SLERATE Community. It guides prospective members through entering essential profile details, accepting Terms of Service and Privacy Policy, and completing email confirmation. Upon successful registration, users gain access to member-only content, channels, and community interactions.

1. Scenarios

|  |  |
| --- | --- |
| **Scenario Name** | **Member Fill in Form, and Submit it via Website** |
| Background | Meena registers for the SLERATE Community by website using her email and completes her sign up by filling in form and submitting it. |
| Objective | Meena must be able to sign up smoothly via website. Mena needs to fill in form and submit it and receive automated confirmation mail |
| Persona(s) | Meena – Member |
| Scenario Steps | 1. Mena is interested in becoming a SLERATE community member. Meena visit SLERATE website in slerate.com > go to the community page > click on the “become a member” button 2. Fill and submit form​. Meena fill in form on <https://slerate.com/community/community-form/>   A screenshot of a computer  AI-generated content may be incorrect.   1. Click submit​ 2. Frontend Validation checks if all required fields are filled. 3. Frontend detects the missing input using client-side validation rules (missing the field with asterisk) 4. The form submission is blocked. 5. The specific empty input field is programmatically targeted and visually highlighted. An inline error message is rendered below or beside the field ("This field is required. Please enter text."). 6. The form remains on the current page until all required fields are completed. 7. No API request is sent to the backend at this point.      1. Meena completed the required field 2. She clicks submit 3. Frontend Checks that all required fields are filled 4. Backend Processing, Input validation    1. Creates new user record in members table with status: pending    2. Triggers email confirmation 5. Frontend response registration successful and UI displays UI displays success screen/message:   A close up of a text  AI-generated content may be incorrect.   1. Auto email confirmation sent. Trigger immediately after successful registration 2. Meena receives an automated email confirmation that registration has been received. |

|  |  |
| --- | --- |
| **Scenario Name** | **Member Information is Processed to Database** |
| Background | After Meena completes and submits the sign-up form on the community app landing page, her data must be securely validated and stored in the members database so that admins/managers can manage the status. |
| Objective | To securely validate and store member registration data into the database and ensure it is accessible to admin (Adam) or manager (Manjit) for approval. |
| Persona(s) | Devina – Developer |
| Scenario Steps | 1. Once validation is complete, the backend controller receives the structured payload. 2. All of the member data is saved into a dedicated database table 3. Once the data is verified and sanitized, the backend service runs an INSERT command to database 4. System Response    1. On Success: Backend returns status “201 Created”.    2. On Failure (duplicate email): Backend returns error (e.g., HTTP 409 Conflict if email already exists). |

|  |  |
| --- | --- |
| **Scenario Name** | **1 level Approval Workflow** |
| Background | Adam logs into the admin dashboard and reviews new community membership applications. He views Meena’s profile submission and approves her request and trigger approval email. |
| Objective | Admin must be able to view pending applications, assess user details, approve or reject the request, and trigger automated onboarding status updates. |
| Persona(s) | Adam – Administrator  Manjit – Manager |
| Scenario Steps | 1. Adam logs into the SLERATE Admin Panel. 2. He clicks the “Member Applications” tab from the sidebar menu. 3. A list of pending applicants is displayed   A screenshot of a computer  AI-generated content may be incorrect.   1. Adam clicks “View” on Meena Kartika’s row to see full application details. The profile view shows all the details from the form   A screenshot of a computer  AI-generated content may be incorrect.   1. Adam clicks the “Approve” button.      1. The system executes the following actions: 2. Updates Meena’s status in database to Approved 3. Sends approval email 4. Adam sees a success message on screen: “Meena Kartika has been approved and notified.” 5. If Meena is rejected, the system executes the following actions:    1. Updates Meena’s status in database to Rejected    2. Sends rejected email 6. Adam repeats the process for the next applicant or returns to dashboard. |

|  |  |
| --- | --- |
| **Scenario Name** | **2 level approval** |
| Background | Members from North America and Africa require an additional approval step: after Admin’s approval, their status remains pending until reviewed by the Manager. This ensures an additional layer of screening for specific regions. |
| Objective | To enable a structured two-level approval process where Admin conducts the first screening and Manager performs final validation for sensitive regions (North America & Africa). |
| Persona(s) | Adam – Administrator  Manjit – Manager |
| Scenario Steps | 1. Adam logs into the Admin Panel. 2. The system displays the full list of member registrations. 3. Adam filters list by Pending status to view only pending member applications. 4. Adam reviews a member from North America or Africa, clicks Approve. 5. System saves approval action but keeps the member status as pending because second-level approval is required. 6. The system notifies Manjit that new pending members from sensitive regions require review. 7. Manjit logs into the Admin Panel. 8. Manjit applies the same Pending filter and sees members already approved by Admin but still pending due to region-based rule. 9. Manjit opens the full profile of the member, including region and submitted data. 10. Take Action:     1. If Approve, Manager clicks Approve → system updates member status to approved.     2. If Reject, Manager clicks Reject → system updates member status to rejected. 11. System Actions:     1. Status updated in database accordingly (approved or rejected)     2. Member receives confirmation email (approval or rejection)     3. Approval actions (Admin + Manager IDs, timestamps) logged in audit trail for traceability. 12. End State: Member is either fully accepted into the community or formally rejected. |

|  |  |
| --- | --- |
| **Scenario Name** | **Members create passwords and login to the app** |
| Background | After Meena is approved, she receives an email prompting her to set up her login credentials. |
| Objective | To enable members to securely create a password and log in to the app using their registered email address, granting them access to member-only features and content. |
| Persona(s) | Meena – Member |
| Scenario Steps | 1. Meena opens the link that sent to her approval email. 2. The link is redirected to the page to set her password for her account in a community app.      1. Meena on the “Set New Password” page enters:    1. New password    2. Confirm password 2. Meena submits the new password form. 3. The system validates the new password:    1. Password must meet criteria (minimum 8 characters, including capital, numbers, and symbol).    2. Password and confirm password must match. 4. If password validation fails, the system shows an appropriate error message and asks Meena to correct inputs.      1. If password validation succeeds, Meena click confirm and the system:    1. The system saves the new password securely.    2. System displays success message: *"Your password has been successfully created. You are now logged in."*      1. Meena click login button and system redirects Meena to the login page      1. She fills in her email and password and clicks login button 2. The system validates her email and password is align, if validation fails, the system shows an appropriate error message and asks the Meena to correct inputs.      1. If validation is successful, she clicks login button and system redirect her into home page community app to accept term of use and privacy policy 2. On the other way, instead of filling a password creation form, the Meena taps either “Sign in with Google” or “Sign in with Apple.” 3. The app initiates the OAuth flow with the chosen provider (Google/Apple). 4. The provider displays a login prompt or account selection to the Meena (outside the app or via embedded WebView). 5. Meena authenticates on the provider’s side and approves permissions requested (typically email and profile). 6. Upon successful authentication, the app receives an authorization token (ID token or access token). 7. The app backend verifies the token with Google or Apple servers to confirm validity and authenticity. 8. Login Success:    1. The app creates an authenticated session for the Meena.    2. Meena is redirected to the app’s dashboard/homepage. 9. A popup automatically appears, containing a summary of the Terms of Use and Privacy Policy, along with links to read the full documents.      1. The popup includes a checkbox labeled, for example: “I have read and agree to the Terms of Use and Privacy Policy.” 2. Meena must check this box to indicate her agreement. 3. Once the box is checked, the “Agree” (or “Continue”) button becomes enabled. 4. Meena clicks the “Agree” button. 5. The system records the agreement in the database/agreement log. 6. Meena now has full access to the application features as a logged-in user. 7. If Meena clicks “Agree” without checking the box, the system blocks the action and displays: “Please check the box to agree before continuing.” 8. If the Terms of Use or Privacy Policy are updated in the future, the popup will reappear for all members to request renewed consent before further use. |

1. Business Requirements

|  |  |  |
| --- | --- | --- |
|  | **Business Requirement** | **Priority** |
| 1 | Ensure users can sign up via website by filling a complete form and submitting it securely. | High |
| 2 | Validate and store all submitted member data in the database with a pending status. | High |
| 3 | Provide a seamless 1-level approval workflow for Admin to review, approve, or reject applicants. | High |
| 4 | Provide a conditional 2-level approval workflow for specific regions (North America & Africa). | High |
| 5 | Allow approved users to set a secure password and access the member app. | High |

1. Functional Requirements

|  |  |  |
| --- | --- | --- |
|  | **Functional Requirement** | **Business Requirement(s)** |
| 1 | Display and validate member registration form on the website. | 1 |
| 2 | Prevent form submission unless all required fields are completed. | 1 |
| 3 | Store validated form data into members database with initial status pending. | 2 |
| 4 | Trigger automated confirmation email after successful submission. | 2 |
| 5 | Enable Admin to view all pending applications and take action to approve or reject. | 3 |
| 6 | Update status to approved or rejected in the database based on Admin input. | 3 |
| 7 | Implement region-based escalation logic that sends certain applications (e.g., from NA or Africa) to the Manager for secondary approval. | 4 |
| 8 | Enable Manager to filter and review escalated applicants and approve or reject them. | 4 |
| 9 | Trigger onboarding or rejection email after final approval decision. | 3, 4 |
| 10 | Allow approved members to log in using their registered email. | 5 |
| 11 | Redirect approved members to create a password upon first login. | 5 |
| 12 | Validate password strength and confirm match before saving. | 5 |
| 13 | Save hashed password securely and authenticate user login. | 5 |
| 14 | Redirect user to app dashboard upon successful login. | 5 |

1. Dependencies

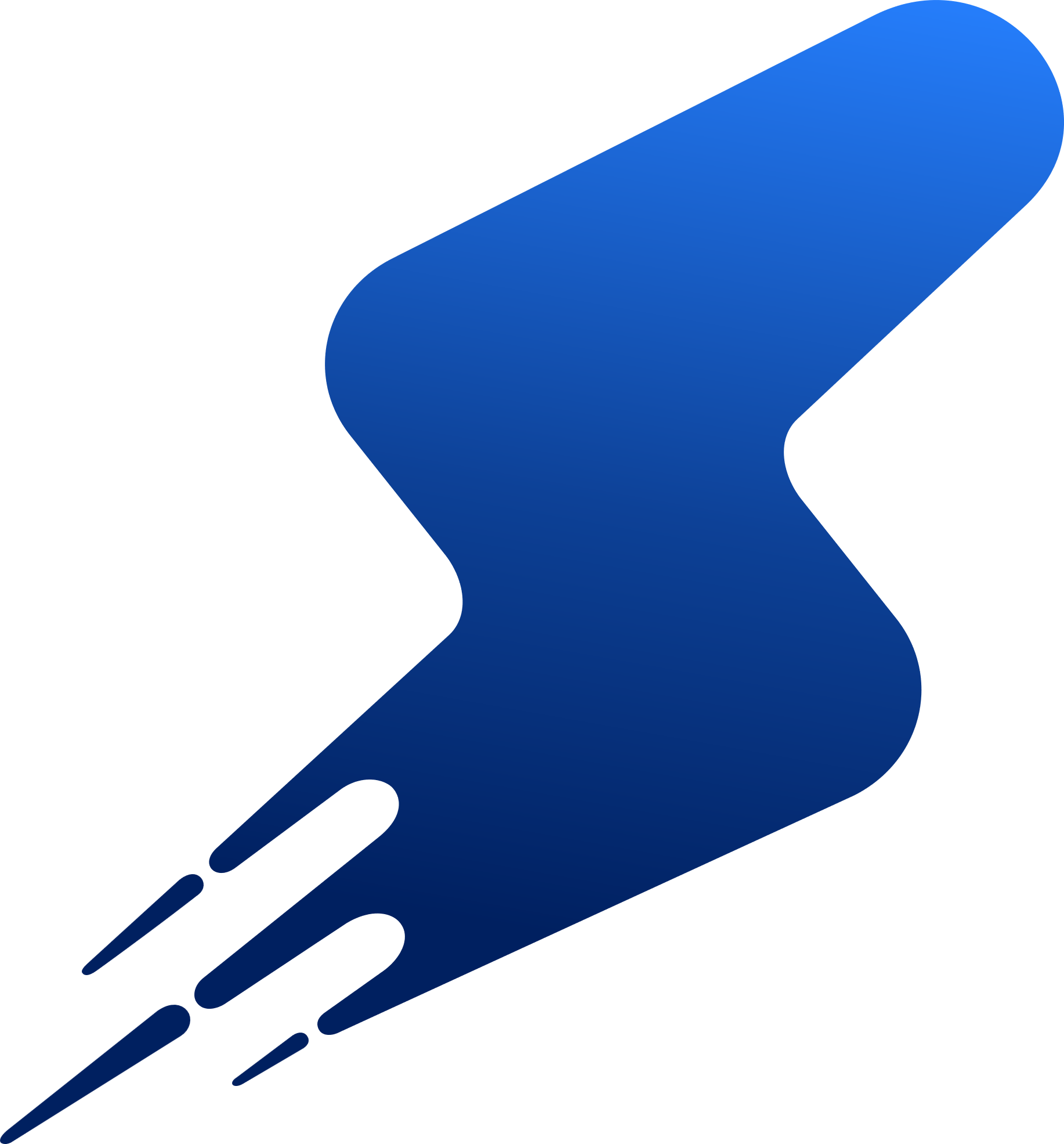
|  |  |  |  |
| --- | --- | --- | --- |
|  | **Team/Contact** | **Type** | **Description** |
| 1 | UI Designer | Technical | Responsible for implementing member sign-up form and form validation UI. |
| 2 | Frontend | Technical | Provides front end program to ensure seamless process |
| 3 | Backend | Technical | Provides API for form submission, database integration, email triggering. |

1. Checklist

|  |  |  |
| --- | --- | --- |
|  | **Checklist Item** | **Response** |
| 1 | Has this functional spec been reviewed by Dev? | No |
| 2 | Has this functional spec been reviewed by UI? | Yes |
| 3 | Has this functional spec been reviewed by Manager? | No |
| 4 | Is the member sign-up form UI fully implemented with required validations? | Yes |
| 5 | Is the API correctly receiving and storing form data in the database? | Yes |
| 6 | Is a confirmation email sent upon successful submission? | No |
| 7 | Can Admin/Manager view and process new member applications in admin panel? | Yes |
| 8 | Are escalated approvals (NA/Africa) correctly handled by Managers only? | Yes |

1. Change and Review history

| **Date** | **Author, editor, or Reviewer** | **Description of change(s)/Comments on review** |
| --- | --- | --- |
| 29/7/2025 | Tahta (Author) | Create the whole Quick specs document |
| 30/7/2025 | Airul (Editor) | Add mockup |
|  |  |  |



SLERATE

July 2025

Copyright © SLERATE

www.SLERATE.com