Team Name: AAA Battery

Project Team Information

Roles	Assigned Members
Project Manager/ Lead	**Abdullah Zubair Ghouri**
Scrum Master	**Aisha Siddiqa**
Requirement Engineer	**All**
Developers	**All**
Testers	**Ayesha Ejaz, Aisha Siddiqa**
UI Designers	**Abdullah Zubair Ghouri**

Sprint 1 Backlog for Library Management System – User Account Management Module

Module Overview

Description:
This module is responsible for managing user accounts, including account registration, email verification, secure login, and password recovery. It serves as the gateway for users to access the library system while ensuring data integrity and security.

Goals:

- Provide a seamless and secure user onboarding process.
- Ensure that all user data is validated, encrypted, and stored securely.
- Set a strong foundation for future modules by implementing robust authentication mechanisms.

User Stories and Acceptance Criteria

1. User Registration

User Story:
As a user, I want to register an account so that I can access the library system.

Acceptance Criteria:

- The registration form must require the user's name, email, and password.
- Input validations must enforce non-empty fields, proper email format, and strong password rules (e.g., minimum 8 characters, inclusion of numbers and special characters).
- Upon successful submission, a verification email is dispatched to the user's email address.
- In case of errors, clear messages guide the user on how to correct the input.

2. Email Verification (Sub-User Story)

User Story:
As a user, I want to verify my email after registration so that I can confirm my identity and complete my account setup.

Acceptance Criteria:

- A unique, secure token is generated during registration and included in the verification email.
- Clicking the verification link validates the token against the system records.
- The user's account status is updated to "verified" upon successful validation.
- If the token is invalid or expired, an error message is displayed with guidance on next steps.

3. User Login

User Story:
As a user, I want to log in securely so that my borrowing history and personal data remain protected.

Acceptance Criteria:

- The login form requires the user's email and password.
- The system only allows login for users with verified email addresses.
- On successful authentication, a secure session is created, and the user is redirected to their dashboard.
- Error messages are displayed if the credentials are incorrect or if the account remains unverified.

4. Password Recovery

User Story:
As a user, I want to recover my password so that I can regain access to my account if I forget it.

Acceptance Criteria:

- A password recovery form is provided to enter the registered email address.
- The system sends a secure, time-limited password reset token to the email.
- Upon clicking the reset link, users are prompted to enter and confirm a new password that meets security criteria.
- The password update is confirmed with a success message, or errors are handled appropriately if the token is invalid or the new password is weak.

Detailed Structured Specifications

Specification 1: User Registration

Input:

- Name

- Email
- Password

Process:

- **Input Validation:**
    - Ensure all fields are filled.
    - Validate email format (e.g., regex check).
    - Check password strength against predefined rules (e.g., length, complexity).
- **Data Handling:**
    - Encrypt the password using secure hashing algorithms (e.g., bcrypt).
    - Store user details in a secure, relational database.
- **Email Dispatch:**
    - Generate a unique verification token.
    - Send an email containing the verification link using a reliable email service.

Output:

- A confirmation message indicating that registration is successful and prompting the user to verify their email.
- Error messages if any validation or storage issues occur.

Specification 2: Email Verification

Input:

- Verification token extracted from the URL in the verification email.

Process:

- **Token Validation:**
    - Check if the token exists in the database and has not expired.
    - Ensure the token corresponds to the correct user account.
- **Account Update:**
    - If the token is valid, update the user's account status to "verified".
    - Invalidate the token after successful verification to prevent reuse.

Output:

- A message confirming successful email verification.
- An error message if the token is invalid or expired, with instructions for requesting a new verification email if needed.

Specification 3: User Login

Input:

- Email
- Password

Process:

- **Credential Authentication:**
    - Validate the email and password combination by comparing the provided password (after encryption) with the stored encrypted password.
    - Verify that the user's email has been confirmed via the email verification process.
- **Session Management:**
    - On successful authentication, generate a secure session token (e.g., JWT or a server-managed session).
    - Maintain session security with proper timeout and renewal policies.

Output:

- A successful login redirect to the user's dashboard, along with a success notification.
- Error messages detailing invalid credentials or an unverified account status.

Specification 4: Password Recovery

Input:

- Registered email address
- (Post-token) New password and password confirmation

Process:

- **Initiation:**
    - Validate the existence of the email in the system.
    - Generate a secure, time-limited password reset token.
    - Dispatch an email containing the reset link and token.
- **Password Reset:**
    - When the user clicks the reset link, prompt them to enter a new password.
    - Validate the new password against security criteria.
    - Encrypt the new password and update the user's record in the database.

Output:

- A confirmation message indicating that the password has been successfully updated.
- Error messages if the email is unregistered, the token is invalid, or the new password does not meet the criteria.

Architecture & Security Considerations

Security Measures:

- All data transmissions use HTTPS to ensure secure communication.
- Passwords are hashed using modern cryptographic techniques before storage.
- Tokens for email verification and password resets are time-limited and securely generated to prevent forgery.
- Implement rate limiting and CAPTCHA on login and registration forms to mitigate brute-force attacks.

System Architecture:

- The system adopts a client-server model with RESTful APIs to manage user account functionalities.
- A dedicated authentication service handles session management and token generation.
- Integration with third-party email services ensures reliable email delivery.

UI/UX Guidelines

User Interface:

- Clean and intuitive forms for registration, login, and password recovery.
- Responsive design ensures usability on mobile devices and desktops.
- Clear call-to-action buttons (e.g., "Register", "Log In", "Reset Password") to guide the user through the process.

User Experience:

- Immediate feedback on form submissions (success or error messages) to improve the user journey.
- Step-by-step instructions provided during the email verification and password recovery processes.
- Minimalistic design to reduce cognitive load and ensure accessibility for all users.

Testing & Quality Assurance

Test Cases:

- **Registration:**
    - Valid input leads to successful account creation and email dispatch.
    - Invalid input triggers the appropriate error messages.
- **Email Verification:**
    - Correct token validates the account.
    - Expired or incorrect token produces an error.
- **Login:**

    - Verified account logs in successfully.
    - Unverified or incorrect credentials display error messages.
- **Password Recovery:**
    - Valid email sends a reset token.
    - Invalid or unregistered email returns an error.
    - Reset process enforces password strength criteria.

Quality Assurance:

- Conduct unit testing for individual functions (e.g., input validation, token generation).
- Execute integration testing to ensure seamless flow across modules.
- Perform security testing to identify and mitigate vulnerabilities.

Dependencies & Assumptions

Dependencies:

- Reliable email service provider for verification and password recovery emails.
- Secure database management system with encryption capabilities.
- Middleware for authentication and session management.

Assumptions:

- Users have valid email addresses for verification.
- The underlying infrastructure supports scalability to handle increasing user loads.
- Development and testing environments replicate production-level security settings.

Risks and Mitigation Strategies

Potential Risks:

- Unauthorized access due to weak password policies.
- Token interception or misuse during the verification or reset process.
- Brute force attacks targeting login endpoints.

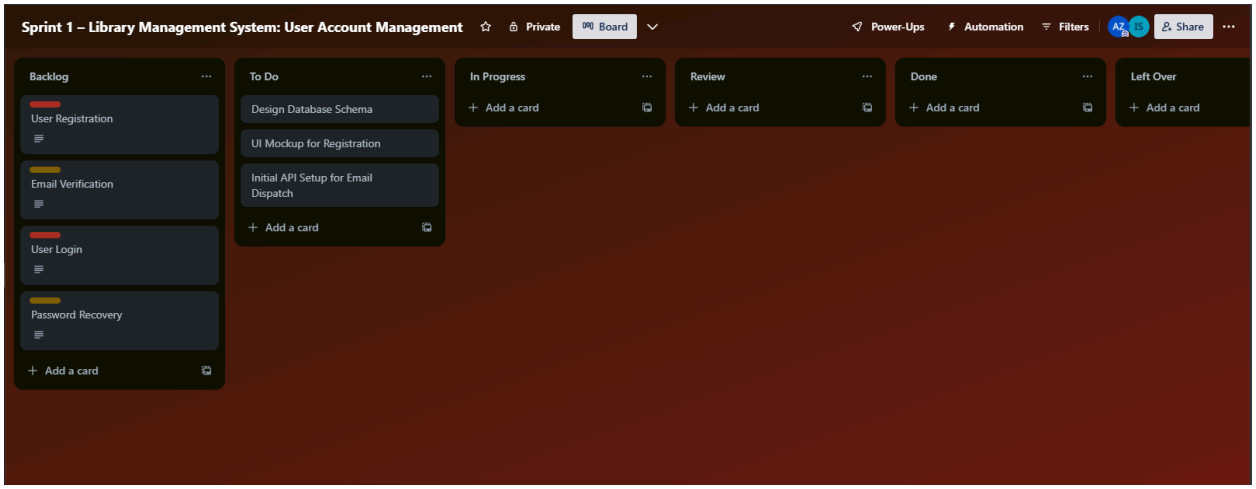
Mitigation Strategies:

- Enforce strong password requirements and regular security audits.
- Use secure, encrypted tokens with limited lifespans.
- Implement rate limiting, CAPTCHA, and two-factor authentication for enhanced security.

Sprint Iterations and Events

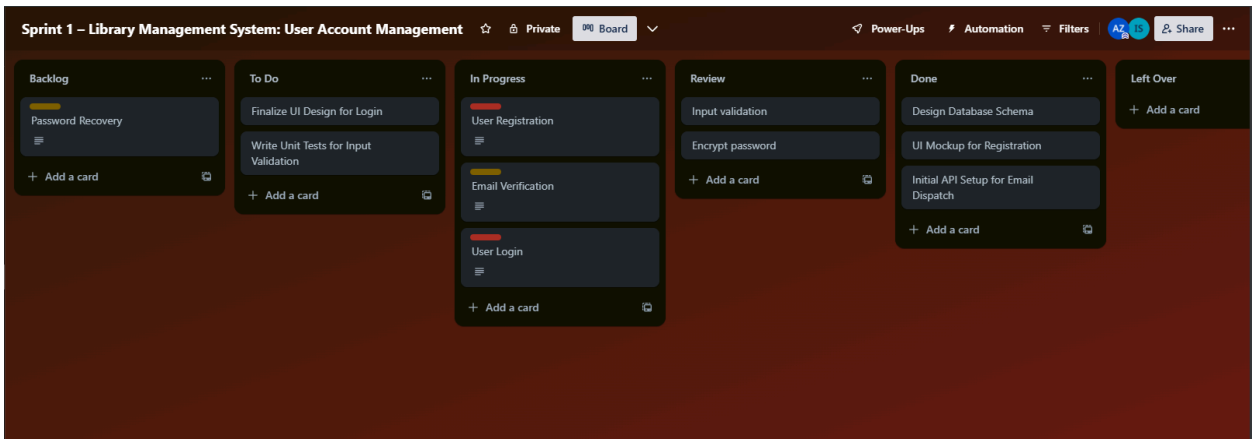
1. Snapshot 1: Initial Board Upload (Sprint Start)

Description:
At the very beginning of Sprint 1, the board is fully populated with all planned tasks and user stories. This snapshot captures the planning phase before any work has begun.



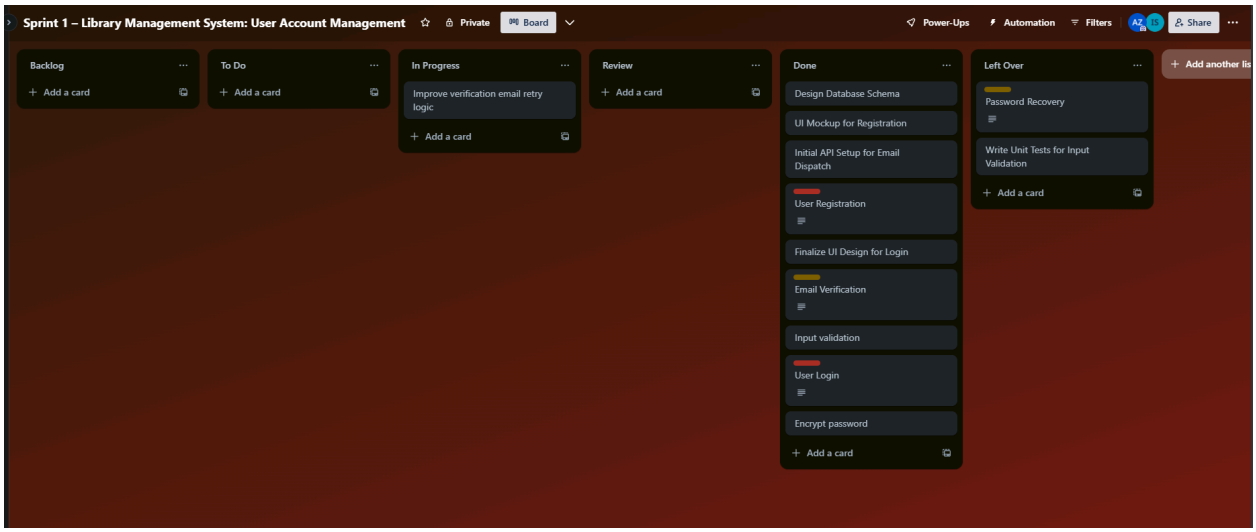
2. Snapshot 2: Mid-Sprint Board (Midway Progress)

Description:
At the midpoint of Sprint 1, progress is evident as tasks begin moving between columns.

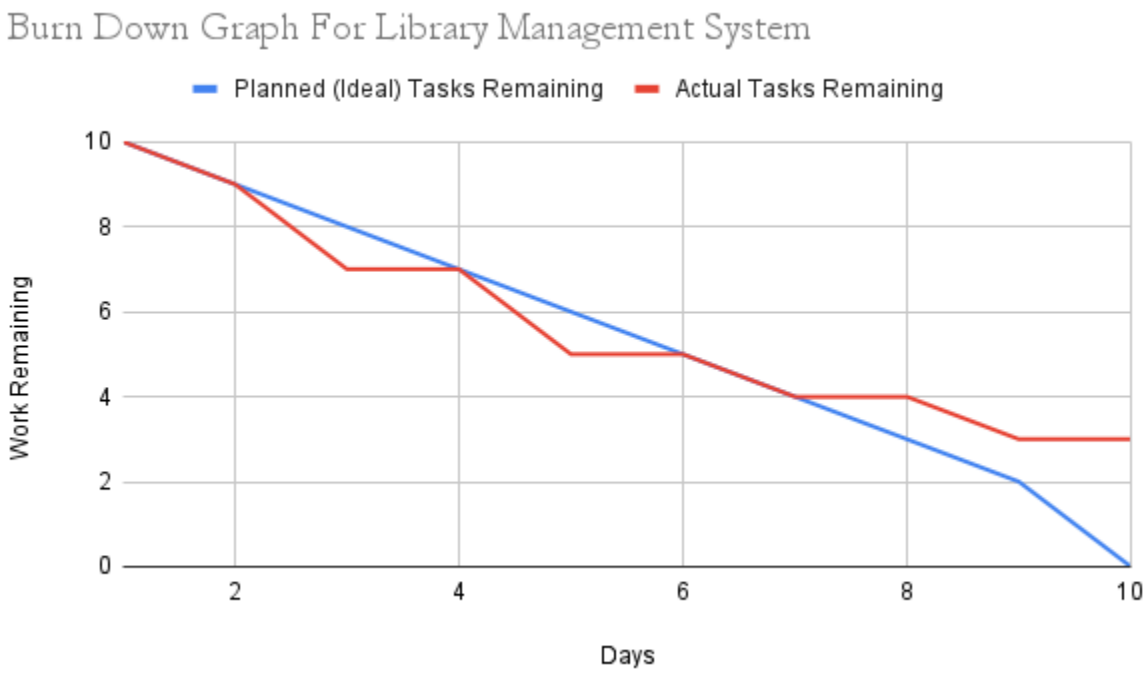


3. Snapshot 3: End-of-Sprint Board (Sprint Completion)

Description:
At the end of Sprint 1, the board reflects the final status of all tasks.



Burndown Chart for above Sprint Cycle



Day	Planned (Ideal) Tasks Remaining	Actual Tasks Remaining	Notes
1	10	10	Sprint start; all tasks in Backlog/To Do.
2	9	9	Minor progress on design/initial setup.
3	8	7	“Design DB Schema,” “UI Mockup,” “Initial API Setup” moved to Done.
4	7	7	Mid-development on “User Registration.”
5	6	5	“Encrypt password” subtask completed, more tasks moved to Review/Done.
6	5	5	Stabilizing progress on “Email Verification,” “User Login.”
7	4	4	Finalizing major tasks; minor tasks remain.
8	3	4	Delays on “Password Recovery” & “Verification Retry Logic.”
9	2	3	“User Login,” “Email Verification,” “User Registration” fully done.
10	0	3	End of Sprint; 3 leftover tasks (“Password Recovery,” “Write Unit Tests,” “Verification Retry Logic”).

Interpreting the Burn-Down Chart

1. **Early Sprint (Days 1–3):**
    - The team starts with all 10 tasks. By Day 3, they’ve completed a few foundational tasks (e.g., database schema, UI mockups), so the actual line dips from 10 to 7.
2. **Mid-Sprint (Days 4–7):**
    - The bulk of the work on “User Registration,” “Email Verification,” and “User Login” is done. Progress stays close to the ideal line.
3. **Late Sprint (Days 8–10):**
    - Some tasks (“Improve verification email retry logic,” “Password Recovery,” “Write Unit Tests”) remain unfinished. The actual line ends at 3 tasks remaining on Day 10, while the planned line is at 0. This indicates leftover work is carried into the next sprint.

4. **Leftover Tasks:**
    - The final snapshot (Snapshot 3) confirms which tasks must move to Sprint 2. These are clearly shown in the **Left Over** column on your Trello board.