

Software Verification and Validation (ENGI-9839-001)

Professor : Raja Abbas

Library Management System (LMS)

Presented by

Priyanka Saha (ID: 202387432)

Sifat Sabrina Rahman (ID: 202286725)

Introduction

- The Library Management System (LMS) is a web-based application designed to streamline core library operations such as book management, loan tracking, and user role handling. It supports two primary roles: **Library Admin** and **Member**, each with distinct permissions and responsibilities. The system emphasizes modularity, security, and scalability.
- This report focuses on the **Verification and Validation (V&V)** processes applied during development. Verification ensures the system is built according to specifications, while validation confirms it meets user expectations. The report documents the testing strategies, findings, and limitations, providing a comprehensive assessment of the system's quality and reliability.

Methodology

Unit Testing — API-Level Validation with Supertest

- **Tool Used:** Supertest (Node.js)

Scope: Isolated backend route testing (Express.js)

- Focused on verifying the correctness of individual backend routes and logic modules in isolation from the rest of the system. These included tests for borrowing books, returning books with fine calculation, user login validation, and book CRUD operations.

Examples:

- Test: Borrowing a book updates availability.
- Test: Returning a book after the due date applies fine.
- Test: Unauthorized user is blocked from admin route.

Methodology

System Testing — *End-to-End Testing with Cypress*

Tool Used: Cypress

Scope: Full user journey across frontend + backend

- Validates complete workflows from a user's perspective in a browser-based environment. It ensured that all parts of the LMS — user interface, API integration, and database interactions — worked as expected together.

Tested Flows:

- Member logs in → browses books → borrows → returns.
- Admin adds a new book → book appears for all users.
- Access control: Members cannot visit /admin routes.

Methodology

Functional testing

Type: Manual Testing for UI

Scope

- **User & Role Management:**
Creating, updating, deleting users; enforcing role-based access for Admin and Member roles.
- **Book Management:**
Adding, editing, deleting books; handling optional fields and file uploads.
- **Borrow & Return Workflow:**
Borrowing available books, returning them, and tracking loan history and fines.
- **Access Control:**
Ensuring that users can only access features permitted by their roles.
- **System Feedback & Validation:**
Verifying that appropriate messages and validations are triggered for both valid and invalid actions.

Methodology

Exploratory Testing

Business District Tour & Guidebook Tour

Approach: Unscripted user exploration of the live app

Tools: Manual browser testing with UI checklist and notes

Metaphors:

- **Business District Tour** – Focused testing of high-traffic, essential components (like borrow/return flow, login, admin dashboard).
- **Guidebook Tour** – Broader exploratory walkthrough to evaluate less-used or secondary features (e.g., uploading images, pagination, form validations).

Evidence of Unit testing method

Performed using [Supertest](#), a lightweight HTTP assertion library used alongside Jest to test the Express.js API endpoints in isolation.

The primary objective of these tests was to validate the correctness of route-level logic for authentication, book management, loan processing, and user operations.

File	Module Tested	Key Actions Tested
auth.test.js	Authentication	User registration and login with token verification
books.test.js	Book management	Book creation and listing
lending.test.js	Lending transactions	Borrowing and returning books
users.test.js	Admin user management	Creating and listing users

This test confirms successful user registration and login, and asserts that a valid JWT token is returned

```
await request(app).post('/api/auth/register').send({
  username: 'alice',
  password: 'pass123',
}).expect(201);

const res = await request(app).post('/api/auth/login').send({
  username: 'alice',
  password: 'pass123',
}).expect(200);

expect(res.body.token).toBeDefined();
```

Unit test for Book Management – [books.test.js](#)

This unit test ensures that an admin can create a book, and the response matches the input data.

```
const resCreate = await request(app)
  .post('/api/books')
  .set('Authorization', `Bearer ${token}`)
  .field('bookId', 'B1')
  .field('title', '1984')
  .field('isbn', '9780451524935')
  .expect(201);

expect(resCreate.body.title).toBe('1984');
expect(resCreate.body.availableCopies).toBe(2);
```

Loan Processing – [lending.test.js](#)

Tests borrowing and returning a book using the member's credentials. Both endpoints are verified.

```
const borrow = await request(app)
  .post('/api/loans/borrow')
  .set('Authorization', `Bearer ${memberToken}`)
  .send({ bookId, dueDate: dueDate.toISOString() })
  .expect(201);

await request(app)
  .put(`/api/loans/return/${borrow.body._id}`)
  .set('Authorization', `Bearer ${memberToken}`)
  .expect(200);
```

System Testing Evidence

For system testing of the Library Management System, the Cypress framework was employed to validate the full user workflow across the frontend and backend, ensuring proper integration between modules.

Tool Used

- Framework: [Cypress](#)
- Purpose: End-to-end testing to simulate real-world user behavior.
- Test File: borrow_return.cy.js
- Scope: Login → Browse Books → Borrow → View Loans → Return

Test Scenario Overview

- The test verifies the core functionality that enables a member to:
- Log into the system via UI.
- Borrow a book listed in the library catalog.
- View their current loans.
- Return the borrowed book successfully.

Operations Flow

Setup Operations (in before() block):

- Logs in or creates an admin user.
- Creates a sample test book (Cypress Book).
- Creates a standard user (testuser) to perform the borrowing actions.

Test Steps (in it() block):

- User logs into the UI.
- Navigates to the Books section.
- Locates the test book and borrows it.
- Receives a UI alert confirming success: "Book borrowed!"
- Navigates to Loans, then returns the book.
- Receives return confirmation: "Book returned successfully"

```
cy.visit('http://localhost:3000/login');
cy.get('input[placeholder="Username"]').type(username);
cy.get('input[placeholder="Password"]').type(password);
cy.get('button[type="submit"]').click();

cy.contains('Books').click();
cy.contains(testBook.title)
  .parents('.book-card')
  .within(() => {
    cy.contains('Borrow').click();
  });

cy.on('window:alert', (text) => {
  expect(text).to.equal('Book borrowed!');
});

cy.contains('Loans').click();
cy.contains(testBook.title)
  .parents('.loan-card')
  .within(() => {
    cy.contains('Return').click();
  });

cy.get('@alert').should('have.been.calledWith', 'Book returned successfully');
```

**You can view the full system test
walkthrough here:**

 **Cypress System Test – Borrow & Return
Flow**

Functional Testing Scenarios

Test case 1: As an Admin, create a new user with valid data

Steps:

- Log in as admin > Navigate to the Manage Users tab
- For the username field, add a username
- For the password field, add a password
- Click on the role dropdown to select the user role
- Hit the Submit button.
- Verify that a new user has been added

Test case 2: As an admin, modify the newly created users

Steps

- Log in as admin > Navigate to the Manage Users tab
- Click on the “Make Member” button.
- Verify that the user has the member role .
- Click on the Reset password button
- Verify that the password has been reset from the system automatically.
- Click on the Delete button.
- Verify that the user has been deleted successfully.
- Verify that all the modifications are updated on the UI as expected.

Functional Test Scenarios

- **Test case 3:** As an Admin, verify that the user management page displays all the listed users.

Steps

- Log in as admin > Navigate to the Manage Users tab
- Verify that the List of all users is displayed on the user management page

- **Test case 4:** As an Admin, add a new book with all fields

Steps

- Log in as admin
- Navigate to Book Management > Add Book > Fill details > Upload files > Save
- Verify that the Book is added and listed

	A	B	C	D	E	F	G	
1	Test Case ID	Description	Preconditions	Test Steps	Expected Result	Type	Feature	
2	TC-URM-01	Create a new user with valid data	Admin is logged in	Navigate to User Management > Add	New user is created and	Functional	User & Role Management	
3	TC-URM-02	View all users	Admin is logged in	Navigate to User Management	List of all users is displayed	Functional	User & Role Management	
4	TC-URM-03	Update a user's role	Admin is logged in, user exists	Navigate to User Management > Select user > Change role > Update	User role is updated	Functional	User & Role Management	
5	TC-URM-04	Delete a user	Admin is logged in, user exists	Navigate to User Management > Select	User is removed from the	Functional	User & Role Management	
6	TC-URM-05	Enforce role-based access	Non-admin user is	Attempt to access User Management	Access is denied with	Functional	User & Role Management	
7	EX-URM-01	Attempt to create a user with an existing			System shows duplicate	Exploratory	User & Role Management	
8	EX-URM-02	Create a user with missing required fields			System shows validation	Exploratory	User & Role Management	
9	EX-URM-03	Assign an invalid or undefined role to a			System handles unknown	Exploratory	User & Role Management	
10	EX-URM-04	Try accessing user management via direct URL as a non-admin			Access is denied	Exploratory	User & Role Management	
11	EX-URM-05	Update a user's role while they are logged			Role change takes effect	Exploratory	User & Role Management	
12	EX-URM-06	Delete a user who has active book loans			System prevents deletion or	Exploratory	User & Role Management	
13	EX-URM-07	View user list with a large number of users			Pagination or performance	Exploratory	User & Role Management	
14	TC-BM-01	Add a new book with all fields	Admin is logged in	Navigate to Book Management > Add Book > Fill details > Upload files > Save	Book is added and listed	Functional	Book Management	
15	TC-BM-02	Add a book with optional fields left blank	Admin is logged in	Repeat TC-BM-01 but leave image/file	Book is added successfully	Functional	Book Management	
16	TC-BM-03	Edit a book	Admin is logged in, book exists	Navigate to Book Management > Select book > Update fields > Save	Book details are updated	Functional	Book Management	
17	TC-BM-04	Delete a book	Admin is logged in,	Navigate to Book Management > Select	Book is removed	Functional	Book Management	

library_test_cases.xlsx - Excel							
Sabrina Rahman SR							
File Home Insert Draw Page Layout Formulas Data Review View Help Foxit PDF Acrobat Tell me what you want to do							
Share							
B9 Assign an invalid or undefined role to a user							
	A	B	C	D	E	F	G
16	TC-BM-03	Edit a book	Admin is logged in, book exists	Navigate to Book Management > Select book > Update fields > Save	Book details are updated	Functional	Book Management
17	TC-BM-04	Delete a book	Admin is logged in, API is available	Navigate to Book Management > Select	Book is removed	Functional	Book Management
18	TC-BM-05	Fetch books via API		Send GET request to /api/books	List of books is returned	Functional	Book Management
19	EX-BM-01	Add a book with a very long title or special			System handles input	Exploratory	Book Management
20	EX-BM-02	Upload a non-image file as a cover image			System rejects invalid file	Exploratory	Book Management
21	EX-BM-03	Upload a very large PDF			System handles file size	Exploratory	Book Management
22	EX-BM-04	Edit a book while another admin is editing			System handles	Exploratory	Book Management
23	EX-BM-05	Delete a book that is currently borrowed			System prevents or warns	Exploratory	Book Management
24	EX-BM-06	Add a book with missing optional fields			System handles null values	Exploratory	Book Management
25	EX-BM-07	Search for books using partial title or			Search returns relevant	Exploratory	Book Management
26	TC-BR-01	Member borrows an available book	Member is logged in, book is available	Navigate to catalog > Click Borrow	Book is marked as borrowed	Functional	Borrow & Return Books
27	TC-BR-02	Member returns a book	Member has borrowed	Navigate to My Loans > Click Return	Book is marked as returned	Functional	Borrow & Return Books
28	TC-BR-03	System calculates fine for overdue return	Book is returned after	Return book after dueDate	Fine is calculated and	Functional	Borrow & Return Books
29	TC-BR-04	Member views loan history	Member is logged in	Navigate to My Loans	Loan history is displayed	Functional	Borrow & Return Books
30	TC-BR-05	Admin views all lending records	Admin is logged in	Navigate to Lending Management	All lending records are	Functional	Borrow & Return Books
31	TC-BR-06	Admin returns book on behalf of member	Admin is logged in, book is borrowed	Navigate to Lending Management > Select record > Return	Book is marked as returned	Functional	Borrow & Return Books
32	EX-BR-01	Borrow a book that has 0 available copies			System prevents borrowing	Exploratory	Borrow & Return Books
33	EX-BR-02	Return a book that was never borrowed			System handles invalid	Exploratory	Borrow & Return Books
34	EX-BR-03	Borrow and return a book multiple times			System maintains data	Exploratory	Borrow & Return Books
35	EX-BR-04	Simulate system clock change to test			System calculates fine	Exploratory	Borrow & Return Books
36	EX-BR-05	View loan history with filters			Filtering and sorting work	Exploratory	Borrow & Return Books
37	EX-BR-06	Admin returns a book that was already			System handles idempotent	Exploratory	Borrow & Return Books
38	EX-BR-07	Member tries to borrow more books than			System enforces borrowing	Exploratory	Borrow & Return Books

Exploratory Testing

Two structured techniques were used for exploratory testing:

Business District Tour

- Explored core features that receive the most user traffic.
- Focus: Authentication, Book Listings, Borrow/Return Flow, User Dashboard.
- Goal: Validate the most “used” user flows under real-life usage simulation.

Guidebook Tour

- Followed predefined documentation or feature descriptions to verify consistency between expected and actual system behavior.
- Focus: Feature instructions in user documentation were followed and verified.
- Goal: Identify mismatches between documentation and implementation.

Testing Areas & Observations:

Mission

- Explore the robustness of the core loan process of a library management system

Charter

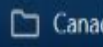
- Login System
- Log in as an existing user with username and password
- Log in as existing user with Google account
- Login as existing user with Facebook account
- Enter incorrect username and password to verify the validation message



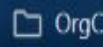
Pers



MUN



Canada St John's



OrgChart



All Bookmarks

[Home](#) | [Register](#) | [Login](#) | [Dashboard](#)

localhost:3000 says

User not found

OK



Login to Your Library Account

adas

.....

Login

Exploratory Testing Tours for User Login

Business District Tour

- Enter valid username and password; verify successful login.
- Try an invalid password or username; check that error messages appear and no access is granted.

Guidebook Tour

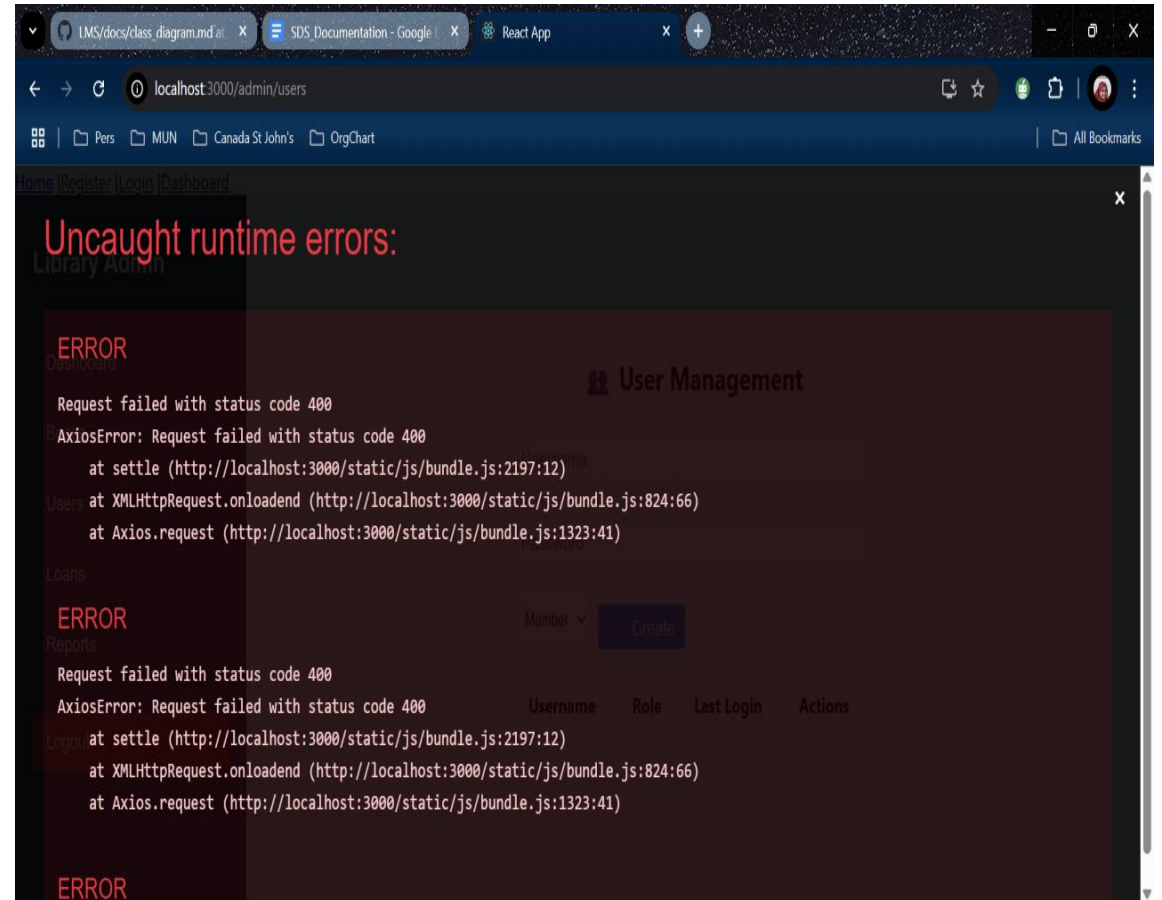
1. Welcome to the Login Portal
2. See input fields for Username or Email and Password
3. Optionally find buttons for Login
4. Users enter their login credentials:
 - Username/Email: Must be valid and registered
 - Password: Should match the stored credentials
5. Typo in email or username
 - CAPS LOCK on when typing password

Authentication Process

- Once users click “Login”:
- The app verifies credentials with the server
- If valid, it creates a secure session (with tokens/cookies)
- If invalid, a friendly error message is shown ("Incorrect password. Try again!")
- Use HTTPS to secure communication
- Implement rate-limiting to prevent brute-force attacks

Bug Report

Session timeout out error occurred while attempting to log in to the system



UI Alignment is broken

[Home](#) | [Register](#) | [Login](#) | [Dashboard](#)


Library Admin

- Dashboard
- Books
- Users
- Loans
- Reports

[Logout](#)


All Loan Transactions (Admin)

Search by Username: Filter by Status:




Test

Test
User: bob
Due Date: 7/26/2025
Returned on 7/19/2025 | Fine: \$0.00



Cypress Book1752947

Cypress Book1752947663338
User: testuser
Due Date: 8/2/2025
Returned on 7/19/2025 | Fine: \$0.00



Cypress Book1752947

Cypress Book1752947514033
User: testuser
Due Date: 8/2/2025
Returned on 7/19/2025 | Fine: \$0.00

Project-Specific Limitations

Limited Scalability Testing

- The system was tested in a controlled environment with limited data and users.
- Load testing (e.g., with concurrent users or large datasets) was not conducted.

Authentication Security Simplification

- Basic JWT-based authentication was implemented.
- Advanced security practices like rate limiting, brute-force protection, and OAuth2 were not integrated.
- This may limit the system's resilience in a production environment.

Research and Documentation Limitations

- No Offline Functionality
- Lack of Automated Reservation Handling Logic
- Research and Documentation Limitations
- Research and Documentation Limitations
- Limited Access to Paid Journals
- Tooling Constraints
- Human Factors in Exploratory Testing

Thank you all !