SABARAGAMUWA UNIVERSITY OF SRI LANKA

LIBRARY MANAGEMENT SYSTEM

PROJECT REPORT

BACHELOR OF SCIENCE
IN
PHYSICAL SCIENCES & TECHNOLOGY

Prepared by,

21APP5704-Fahadh Muhammadh

Prepared for,

Computer Lab Module

Faculty of Applied Sciences

Department of Physical Science Technology

CONTENTS

Problem Overview

Manual book tracking in small libraries often results in misplaced records, overdue confusion, and slow lookup processes. To address this, the project introduces a desktop-based system that centralizes book lending and reduces paper-based errors. The system also includes an authentication mechanism to prevent casual misuse. Currently, login validation relies on a hard-coded admin credential (“Admin” / “Fhd@2948”) inside the main form code, and the initial implementation is built using Windows Forms to function on existing PCs without requiring web hosting.

The project is maintained in the GitHub repository located at *C:\Users\FHD29\Desktop\The Library Management System*, synced with the remote branch at *github.com/FahadhCodes/LibraryManagementSystem*, and targets the .NET 8 framework to ensure modern runtime compatibility.

Design Summary

The system begins with a modal login interface designed using Windows Forms. The layout includes labels, username and password text boxes, and a login button arranged within a 572×290 client area. Controls are anchored to maintain alignment during resizing. For usability, the password field masks input using the ‘•’ character, and the interface features a centered header for clarity.

Authentication occurs when the login button is clicked; the form verifies the provided credentials and returns DialogResult.OK upon success. Errors trigger MessageBox alerts, enabling clear user feedback. The form's icon is embedded directly in resource files (.resx) for consistent packaging.

At the current stage, the application does not include persistent data storage. No book, member, or loan models exist yet. The solution relies solely on UI logic, and the system uses the standard WinForms disposal pattern to release resources when forms close.

Implementation Challenges

Several implementation challenges were encountered during development. Creating the authentication step required a simple yet reliable method, which was achieved by validating login credentials in the LoginForm and returning DialogResult.OK on success. Password protection was provided using the built-in masking property. Ensuring the layout remained stable during window resizing was resolved through the strategic use of anchor properties.

To enhance presentation, an application icon was embedded in resource files, removing dependency on external assets. Clear user feedback was implemented through MessageBox.Show with an error icon to alert users of incorrect credentials. Resource management was handled automatically using the designer-generated Dispose method.

However, secure authentication and persistent data storage remain unimplemented. Storing hard-coded credentials is unsuitable for production, and this limitation is acknowledged as an area for future improvement.

Future Enhancements

Future development will expand the application into a fully functional library management system. Database integration—using SQLite or SQL Server—will introduce permanent storage for Books, Members, and Loans, along with structured schemas and migrations. Authentication can then be upgraded to secure storage with hashed passwords, and role-based access (Admin, Librarian, Member) can be implemented.

Additional improvements include CRUD interfaces for core entities, advanced business logic such as overdue calculations, and reporting features with PDF or CSV export. Enhancing usability through responsive design, validation, localization, and improved accessibility will make the system more robust. Application packaging, automated backups, and unified settings support will improve operational reliability.

On the engineering side, automated testing and continuous integration can be introduced through unit tests and GitHub Actions pipelines. These practices will increase code stability and ensure consistent builds whenever changes are pushed to the repository.