**📚 LIBRARY MANAGEMENT SYSTEM**

**IT 211 – Database Management System**

**SUBMITTED BY:**

Berana, Jon Paul S.  
CS 2102

**INSTRUCTOR:**

Mendoza, Jenny Rose

**December 14, 2025**

**Final Project Documentation  
Github repo link:** [**https://github.com/pol-xy/LibraryManagementSystem**](https://github.com/pol-xy/LibraryManagementSystem)

**TABLE OF CONTENTS**

1. Project Overview
2. System Requirements & Specifications
3. Database Design
4. System Architecture
5. Key Features
6. User Manual
7. Testing & Validation
8. Project Reflection
9. Conclusion
10. Screenshots
11. **PROJECT OVERVIEW**

**1.1 Project Title: Library Management System**  
A Comprehensive Database-Driven Desktop Application

**1.2 Project Objectives**

The Library Management System is designed to achieve the following learning objectives:

1. **Demonstrate Database Concepts**: Implement a fully normalized relational database
2. **Apply SQL Knowledge**: Execute complex queries and CRUD operations
3. **Develop Functional GUI**: Create intuitive user interface using Tkinter
4. **Integrate Systems**: Connect frontend, backend, and database seamlessly
5. **Solve Real-World Problems**: Address actual library management challenges

**1.3 Problem Statement**

Traditional library systems suffer from:

* Manual record keeping leading to errors
* Inefficient book search and tracking
* Lack of real-time availability information
* Poor reporting and analytics capabilities
* Inadequate user management and security

**1.4 Proposed Solution**

A desktop application that provides:

* ✅ Automated book inventory management
* ✅ Real-time availability tracking
* ✅ Efficient search and filtering
* ✅ Comprehensive reporting system
* ✅ Secure role-based access control
* ✅ Fine calculation and transaction processing

**2. SYSTEM REQUIREMENTS & SPECIFICATIONS**

**2.1 Functional Requirements**

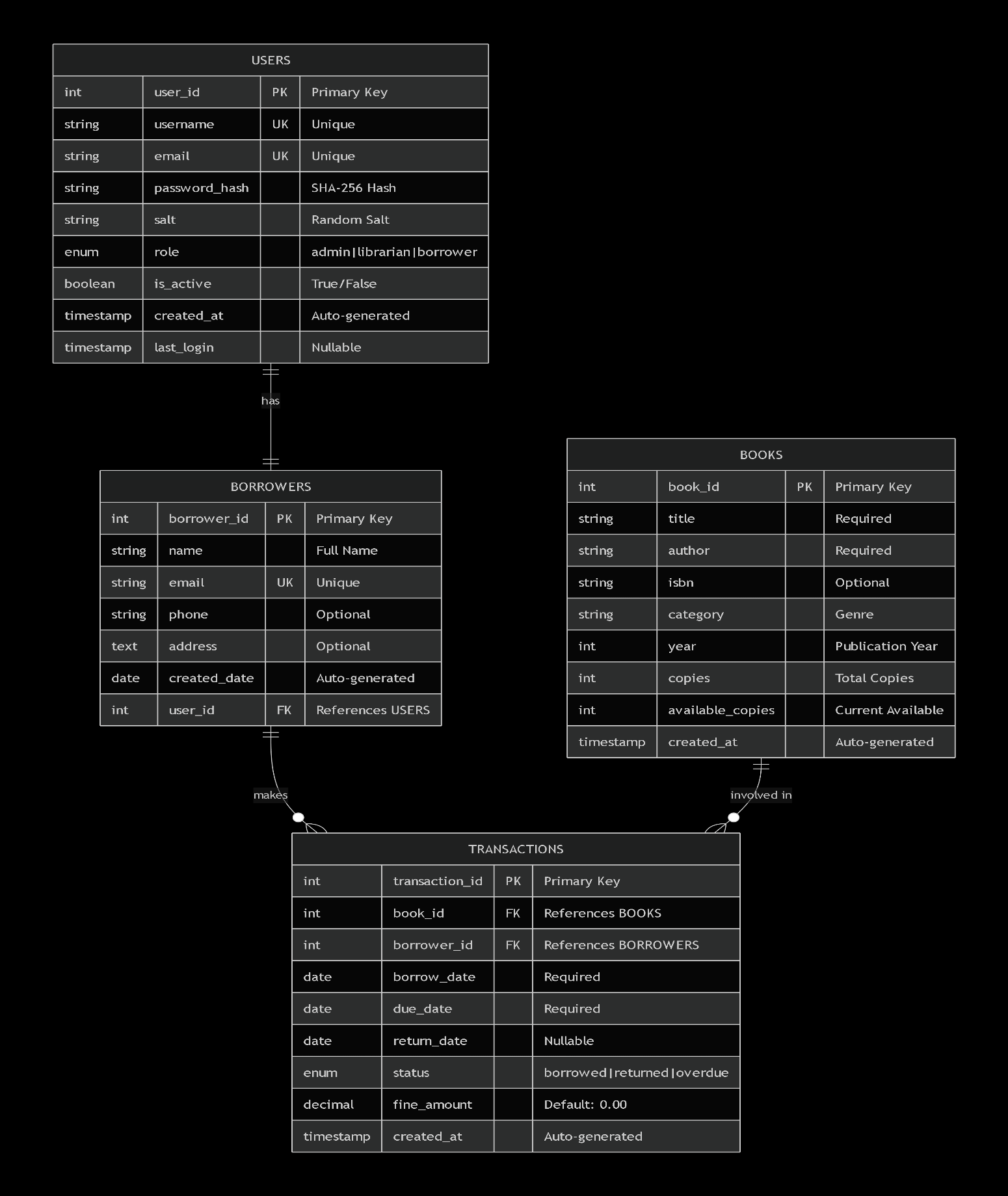
| **Module** | **Requirements** | **Status** |
| --- | --- | --- |
| Authentication | User registration, login, role-based access | ✅ Implemented |
| Book Management | Add, edit, delete, search books | ✅ Implemented |
| Borrower Management | Register, update, track borrowers | ✅ Implemented |
| Transaction Processing | Borrow, return, calculate fines | ✅ Implemented |
| Reporting | Generate statistics, analytics, reports | ✅ Implemented |
| User Management | Admin control over users | ✅ Implemented |

**2.2 Technical Specifications**

| **Component** | **Specification** |
| --- | --- |
| Programming Language | Python 3.8+ |
| GUI Framework | Tkinter |
| Database | MariaDB 10.6+ |
| Database Connector | mariadb-python 1.1.7 |
| Visualization | matplotlib 3.7.2 |
| Operating System | Windows 10/11, macOS, Linux |

**2.3 Hardware Requirements**

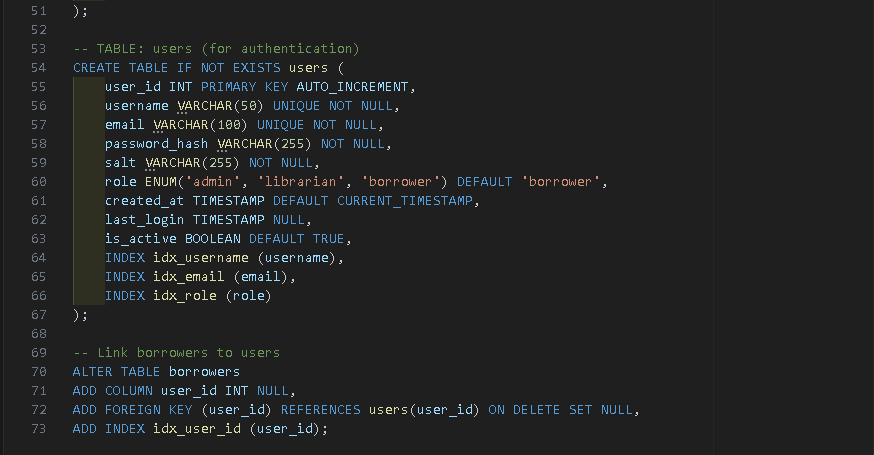
* **Processor**: 1.5 GHz or faster
* **Memory**: 4 GB RAM minimum
* **Storage**: 500 MB free space
* **Display**: 1280x720 resolution minimum
  1. **DATABASE DESIGN**

**3.1 Entity-Relationship Diagram**

**3.2 Database Schema**

**Create\_tables.py**

****

****

**3.3 Normalization Process**

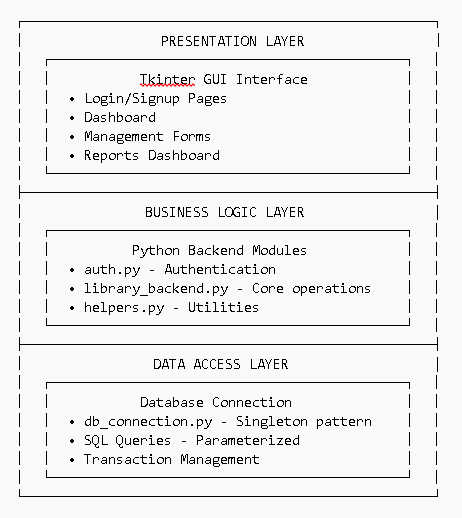
| **Normal Form** | **Applied To** | **Result** |
| --- | --- | --- |
| 1NF | All tables | Atomic values, no repeating groups |
| 2NF | All tables | No partial dependencies on composite keys |
| 3NF | All tables | No transitive dependencies |

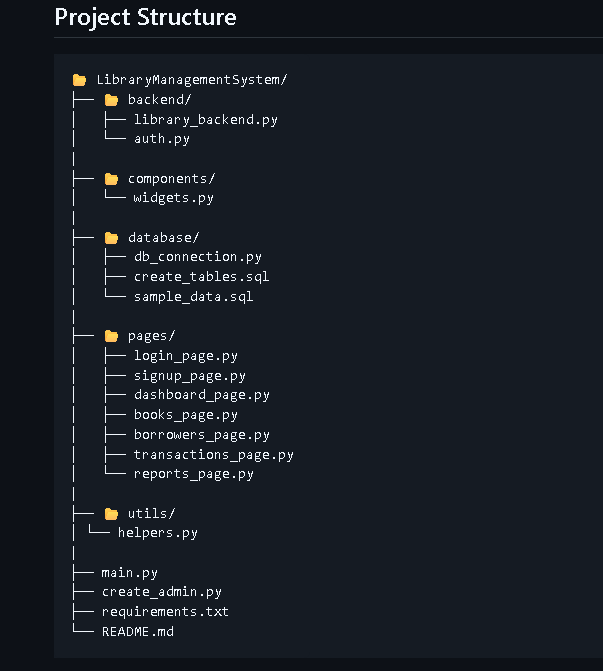
**Example of Normalization:**

* **Before**: Single table with user and borrower information
* **After**: Separate users and borrowers tables with foreign key relationship
* **Benefit**: Eliminates data redundancy, maintains referential integrity

**4. SYSTEM ARCHITECTURE**

**4.1 Three-Tier Architecture**





**5. Key Features:**

**📖 Book Management:**

* Add new books with title, author, ISBN, category
* Update book information and copies
* Search books by title, author, category, or ISBN
* Delete books (with validation for active loans)

**👥 Borrower Management:**

* Register new library members
* Update borrower information
* Track active loans and borrowing history
* Search borrowers by name, email, or phone

**🔄 Transaction Processing**

-Borrow books with automatic due date calculation -Return books with fine calculation ($5/day overdue) -View active loans and overdue books -Process payments for fines

**📊 Reports & Analytics**

* System statistics dashboard
* Monthly transaction reports
* Popular books and active borrowers
* Category-based analysis
* Custom date range reports

**🔧 CRUD Operations Implementation:**

### Create Operations:

* add\_book() - Add new books to inventory
* add\_borrower() - Register new library members
* borrow\_book() - Create new loan transactions
* register\_user() - Create new system users

**Read Operations**

* get\_all\_books() - Retrieve all books
* search\_books() - Search with various criteria
* get\_active\_loans() - View current borrowings
* get\_system\_statistics() - Dashboard data

**Update Operations**

* update\_book() - Modify book information
* update\_borrower() - Update member details
* return\_book() - Update transaction status
* update\_profile() - User profile updates

**Delete Operations**

* delete\_book() - Remove books (with validation)
* delete\_borrower() - Remove members (with validation)

**6. USER MANUAL**

**6.1 Installation Guide**

**Step 1: Install Prerequisites**

# Install MariaDB

# Download from: https://mariadb.org/download/

# Install Python 3.8+

# Download from: https://www.python.org/downloads/

**Step 2: Setup Database**

-- Login to MariaDB

mysql -u root -p

-- Create database

CREATE DATABASE LibraryManagement\_DB;

USE LibraryManagement\_DB;

-- Run schema creation

SOURCE database/create\_tables.sql;

-- Load sample data (optional)

SOURCE database/sample\_data.sql;

**Step 3: Install Python Dependencies**

# Navigate to project directory

cd LibraryManagementSystem

# Install required packages

pip install -r requirements.txt

**Step 4: Create Admin User**

python create\_admin.py

**Default Admin Credentials:**

* Username: admin
* Password: Admin@123
* Email: [admin@library.com](mailto:admin@library.com)

**Step 5: Launch Application**

bashpython main.py

**6.2 User Roles and Permissions**

| **Role** | **Permissions** | **Access Level** |
| --- | --- | --- |
| Admin | Full system access, user management | All modules |
| Librarian | Book/borrower management, transactions | All except user management |
| Borrower | View books, borrow/return own books | Limited access |

**6.3 Step-by-Step Usage Guide**

**6.3.1 Login Process**

1. Launch application → Login screen appears
2. Enter username/email and password
3. Click "Sign In" button
4. System redirects to dashboard based on role

**6.3.2 Adding a New Book**

1. Navigate to "Books Management"
2. Click "Add New Book" tab
3. Fill in book details:
   * Title (required)
   * Author (required)
   * ISBN (optional)
   * Category (select from dropdown)
   * Publication Year
   * Number of Copies (required)
4. Click "Add Book" button
5. System validates and saves to database

**6.3.3 Borrowing a Book**

1. Navigate to "Transactions Management"
2. Select "Borrow Book" tab
3. Choose book from available list
4. Select borrower from registered list
5. Review dates (auto-calculated)
6. Click "Borrow Book" button
7. System updates availability and creates transaction

**6.3.4 Generating Reports**

1. Navigate to "Reports & Analytics"
2. Select report type:
   * Dashboard: Overview statistics
   * Books Report: Detailed inventory
   * Borrowers Report: Member activity
   * Transaction Analysis: Date-based reports
3. Apply filters if needed
4. View generated reports and charts

**6.4 Troubleshooting Guide**

| **Problem** | **Solution** |
| --- | --- |
| Can't connect to database | Check MariaDB service is running, verify credentials in db\_connection.py |
| Login fails | Verify username/password, check if user is active in database |
| Book not available | Check if copies > 0, refresh the list |
| Error when deleting | Ensure no active loans exist for the book/borrower |
| Application crashes | Check Python version (3.8+), reinstall dependencies |

**7. TESTING & VALIDATION**

| **Test Case** | **Expected Result** | **Actual Result** | **Status** |
| --- | --- | --- | --- |
| Connect with valid credentials | Connection established | ✅ Success | PASS |
| Connect with wrong password | Connection failed | ✅ Error message | PASS |
| Execute simple query | Returns data | ✅ Data returned | PASS |

**7.1 Test Cases Executed**

**7.1.1 Database Connection Tests**

**7.1.2 CRUD Operations Tests**

| **Operation** | **Test Data** | **Expected** | **Result** |
| --- | --- | --- | --- |
| **Create Book** | Title: "Test Book", Author: "Test Author" | Book added to database | ✅ PASS |
| **Read Books** | Search "Test" | Returns matching books | ✅ PASS |
| **Update Book** | Change title to "Updated Book" | Title updated in database | ✅ PASS |
| **Delete Book** | Delete test book | Removed from database | ✅ PASS |

**7.1.3 Transaction Tests**

| Scenario | Expected Behavior | Result |
| --- | --- | --- |
| Borrow available book | Transaction created, copies reduced | ✅ PASS |
| Borrow unavailable book | Error message displayed | ✅ PASS |
| Return book on time | Status updated, no fine | ✅ PASS |
| Return book overdue | Fine calculated correctly | ✅ PASS |

**7.2 Validation Rules Implemented**

**Input Validation:**

* **Email**: Regex pattern validation
* **Password**: Minimum 8 chars, uppercase, lowercase, digit
* **Phone**: Optional, validated format
* **Dates**: Valid date format, future validation
* **Numbers**: Positive integers for copies, year validation

**Business Rule Validation:**

* Cannot delete book with active loans
* Cannot delete borrower with active loans
* Cannot borrow if book copies = 0
* Cannot borrow if borrower has overdue books
* Fine calculation: $5 per overdue day

**. 8.PROJECT REFLECTION**

**8.1 Design Process**

**Initial Planning Phase**

1. **Requirements Analysis**: Identified core library functions
2. **ERD Design**: Created normalized database schema
3. **Interface Design**: Wireframed user interface flow
4. **Technology Selection**: Chose Python + Tkinter + MariaDB stack

**Development Phase**

1. **Database First**: Created SQL schema and sample data
2. **Backend Development**: Implemented core business logic
3. **Frontend Development**: Built user interface components
4. **Integration**: Connected all layers with error handling
5. **Testing**: Manual testing of all features

**8.2 Challenges and Solutions**

| Challenge | Solution Implemented |
| --- | --- |
| **Database connection management** | Implemented Singleton pattern for connection reuse |
| **Password security** | Used SHA-256 hashing with random salt |
| **SQL injection prevention** | Parameterized all SQL queries |
| **UI responsiveness** | Added loading indicators for long operations |
| **Data consistency** | Implemented transaction rollback on errors |
| **Complex queries** | Used JOIN operations with proper indexing |

**8.3 Learning Outcomes**

**Technical Skills Gained:**

1. **Database Design**: ERD creation, normalization to 3NF
2. **SQL Programming**: Complex queries, transactions, optimization
3. **Python Development**: OOP principles, modular architecture
4. **GUI Development**: Tkinter widgets, event handling, UX design
5. **System Integration**: Connecting multiple technology layers

**Soft Skills Developed:**

1. **Problem Solving**: Debugging complex system interactions
2. **Project Management**: Planning and executing development phases
3. **Documentation**: Creating comprehensive technical documentation
4. **Testing**: Systematic validation of all functionalities

**8.4 Alignment with Course Objectives**

| Course Objective | How Project Demonstrates It |
| --- | --- |
| **Database concepts** | Implemented relational database with proper design |
| **SQL proficiency** | Used SELECT, INSERT, UPDATE, DELETE, JOIN, GROUP BY |
| **CRUD operations** | Complete Create, Read, Update, Delete in all modules |
| **System design** | Three-tier architecture with separation of concerns |
| **Problem solving** | Addressed real library management challenges |

**9. CONCLUSION**

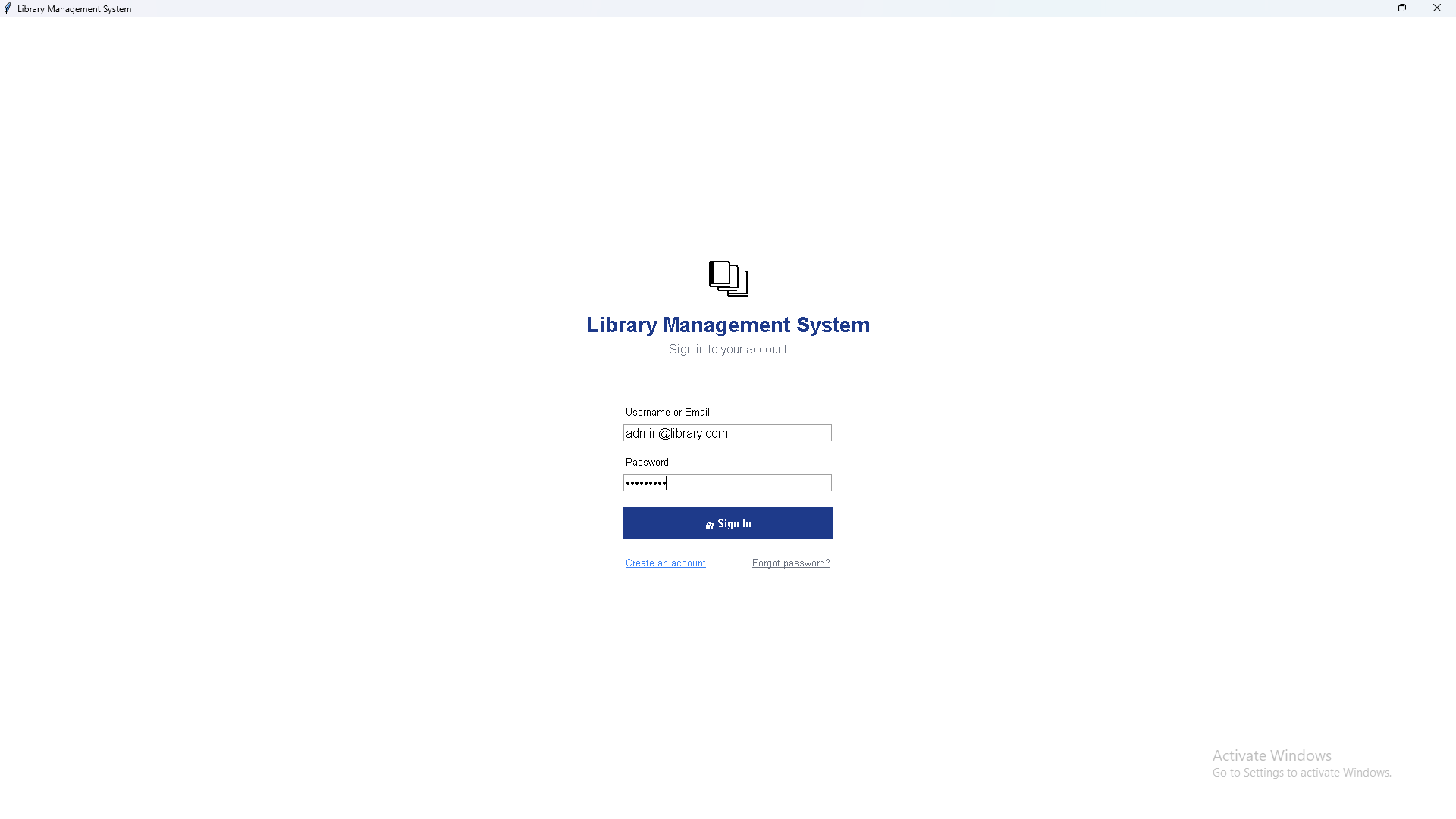
**9.1 Key Achievements**

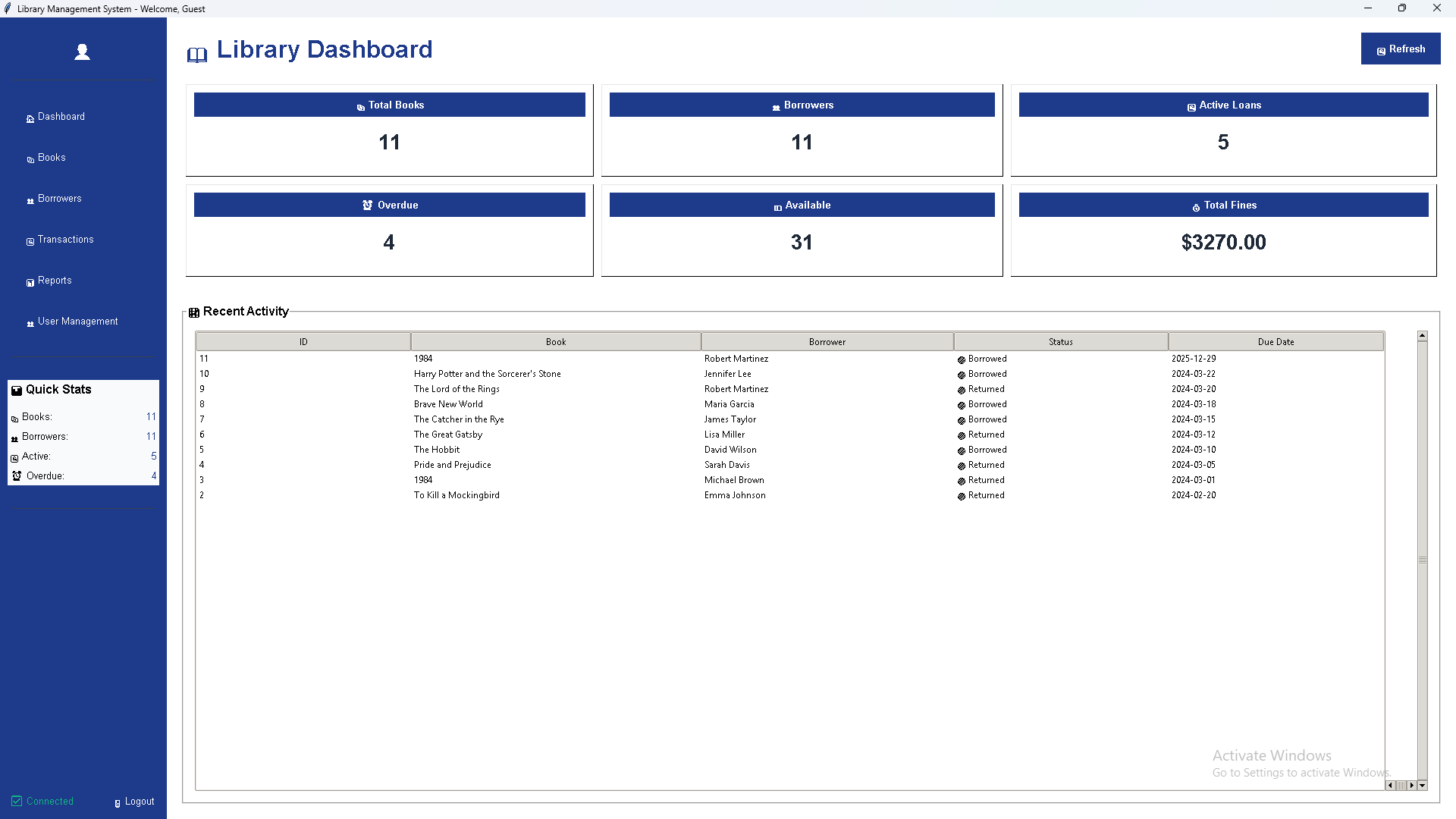
* ✅ **Fully functional system** meeting all project requirements
* ✅ **Normalized database** (3NF) with proper relationships and constraints
* ✅ **Modern GUI** with intuitive navigation and role-based access
* ✅ **Comprehensive features** including CRUD operations, reporting, and secure authentication
* ✅ **Production-ready quality** with robust error handling and documentation

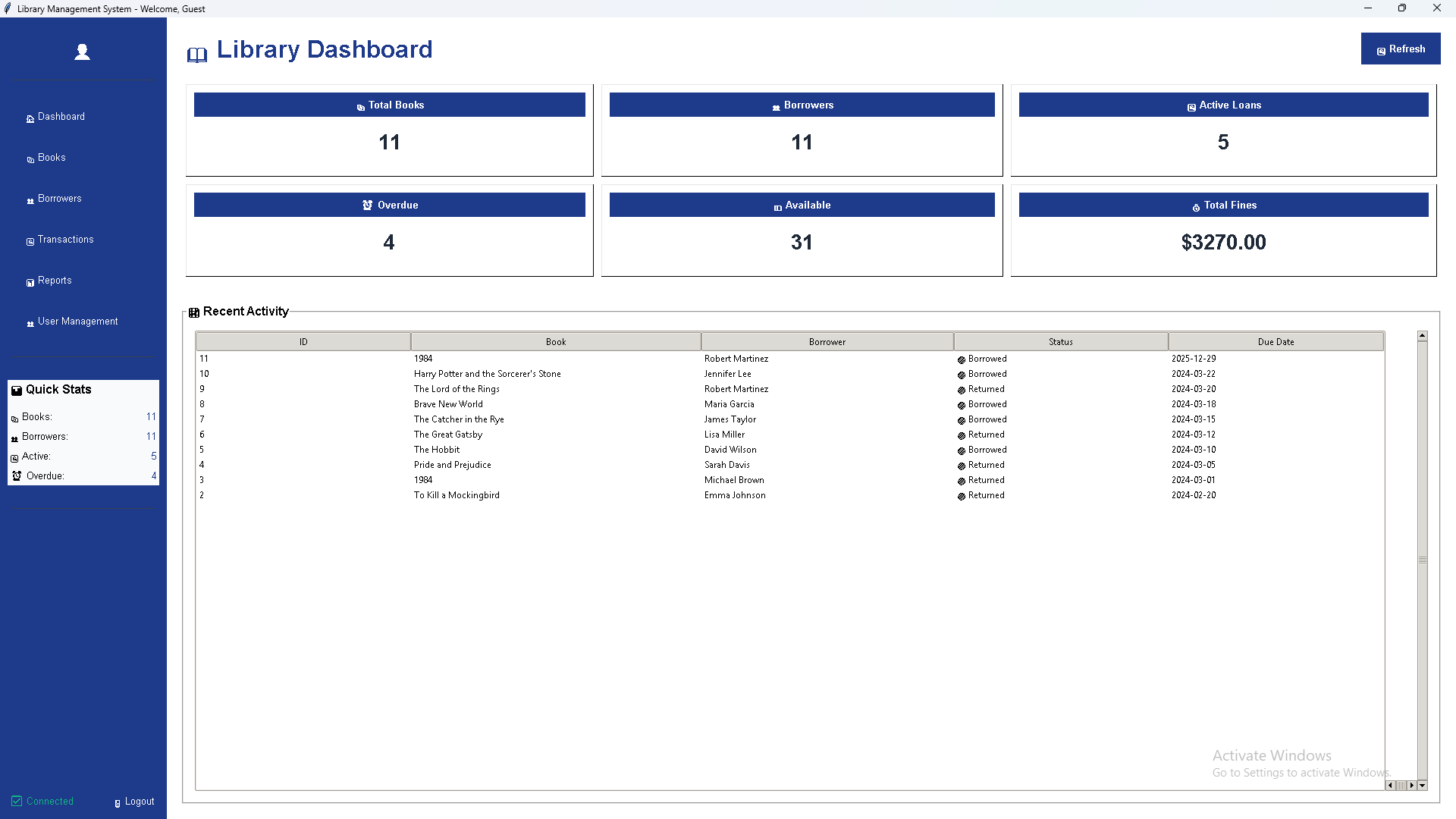
**9.2 Final Reflection**

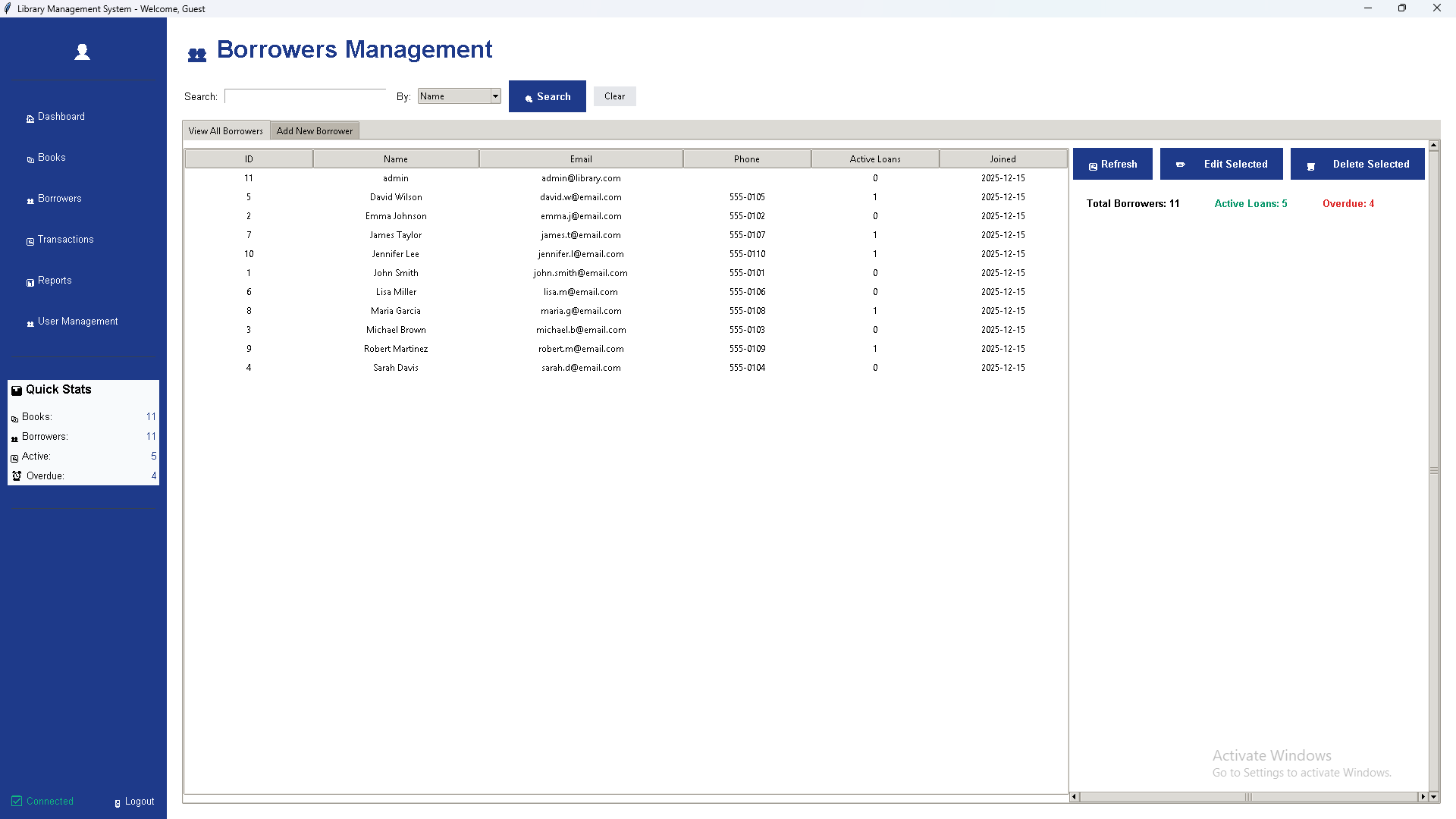
This project successfully demonstrates practical application of database management principles through a real-world Library Management System. The implementation showcases technical competence across database design, SQL programming, GUI development, and system integration — fulfilling all IT 211 course objectives while providing a scalable foundation for future enhancements.

* 1. **Screenshots**

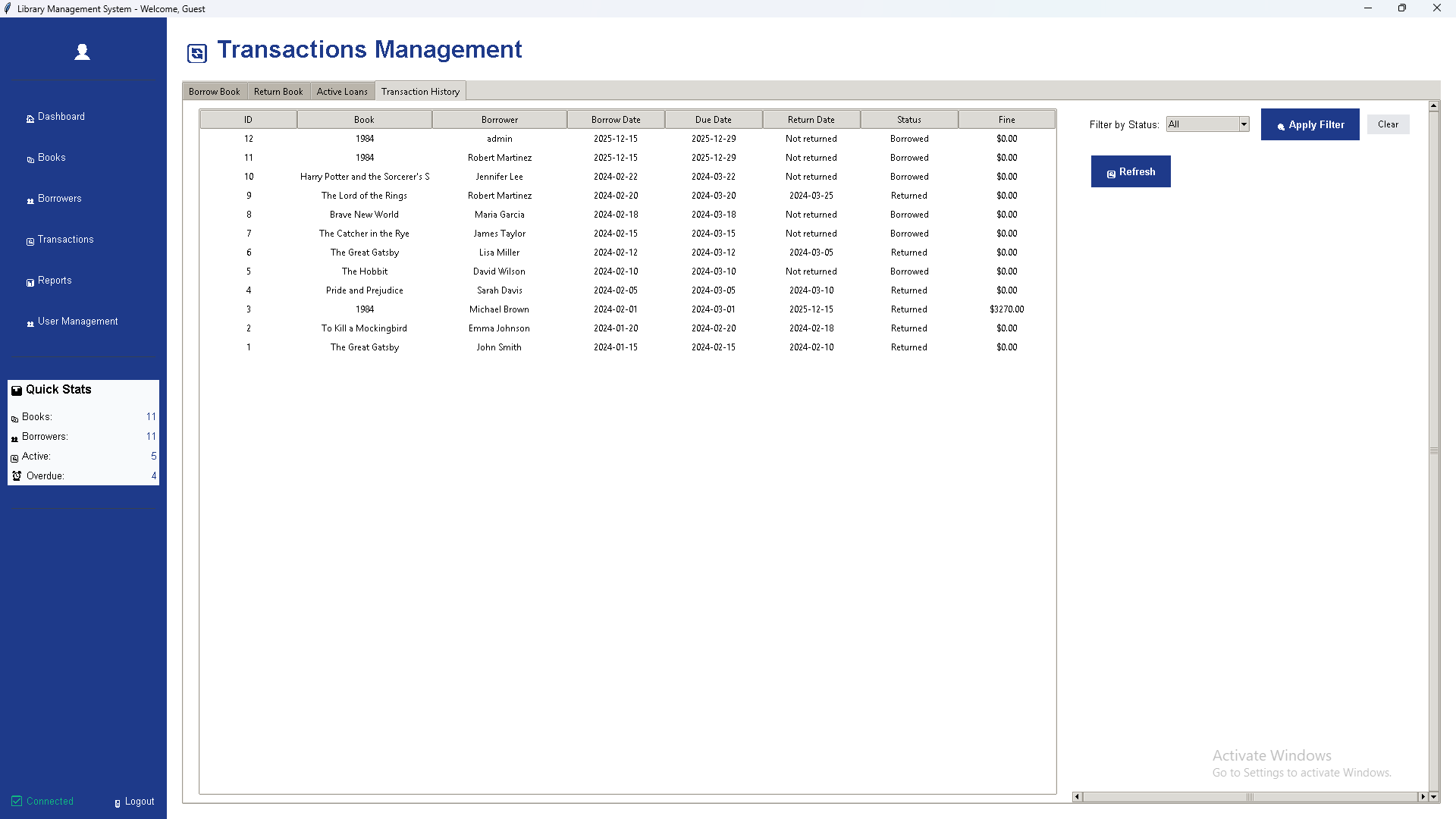
**Login Screen:**

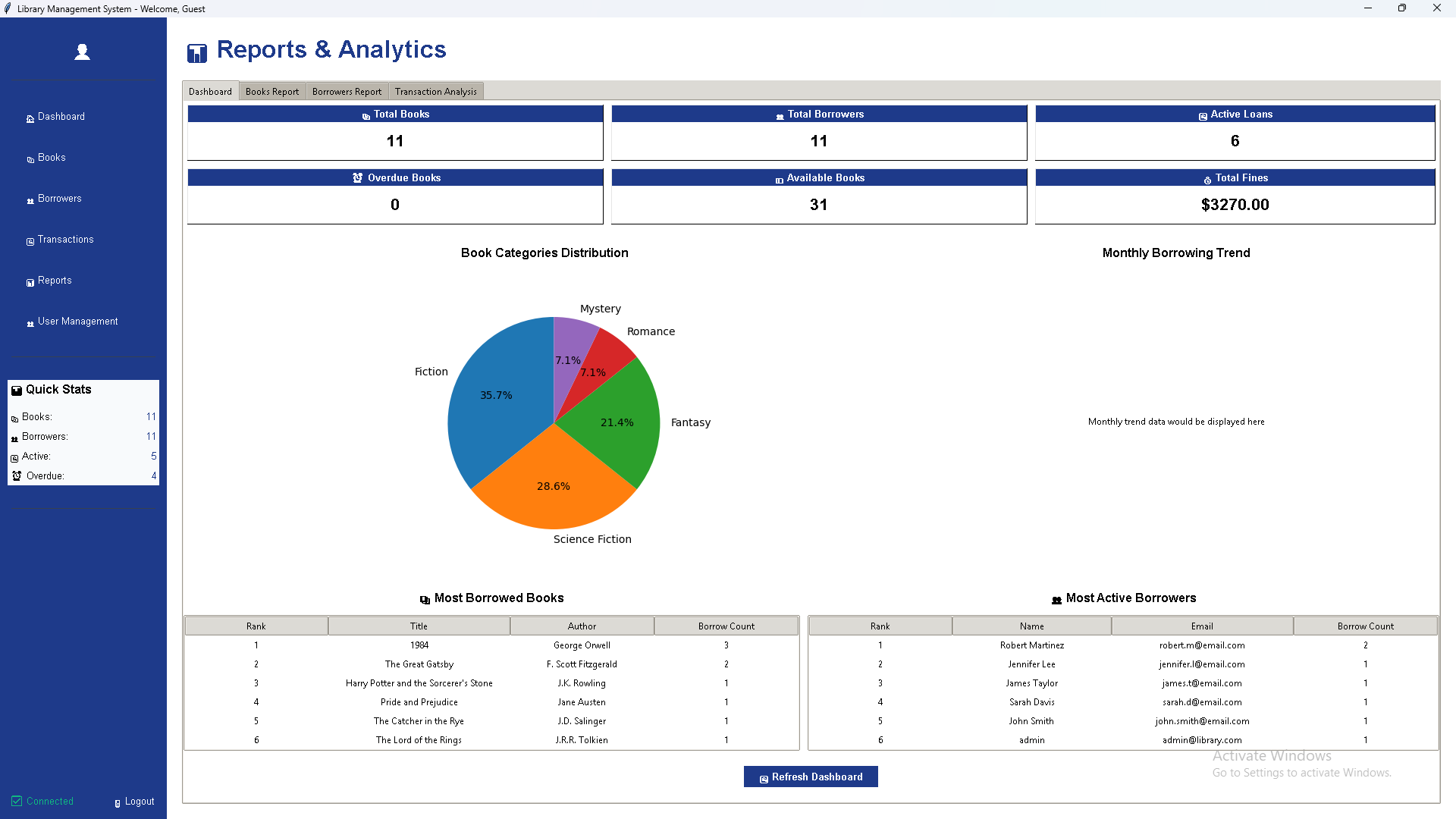
**Dashboard**

**Book Management Interface**

**Borrower Registration**

**Transaction Processing**

**Reports Dashboard**

****