# 📘 Library Management System

**Windows Forms Application – C#**

---

## 1️⃣ Project Overview

This project is a **Library Management System** developed using **C# Windows Forms (WinForms)**. The system allows users to manage:

- Categories
- Books
- Members
- Loans
- Reports

The application follows a **CRUD-based architecture** and connects to a relational database to store and retrieve data efficiently.

---

## 2️⃣ Technologies Used

- **C# (.NET Framework / .NET 8 WinForms)**
- **Visual Studio**
- **SQL Database**
- **ADO.NET**
- **Windows Forms (GUI)**

---

## 3️⃣ Database Design & Migration

Initially, the database schema was designed for **MySQL**, which caused several incompatibilities when integrating with **SQL Server**.

✔ **Fixed SQL Server schema incompatibility with MySQL**

- Converted MySQL-specific syntax to SQL Server–compatible syntax.
- Replaced unsupported data types and keywords.

✔ **Fixed IDENTITY, NVARCHAR, BIT incompatibilities**

- Replaced AUTO_INCREMENT (MySQL) with IDENTITY(1,1) (SQL Server).
- Replaced VARCHAR with NVARCHAR to support Unicode characters.
- Converted TINYINT(1) to BIT for boolean values.

✔ **Ensured foreign key integrity using InnoDB**

- Ensured proper foreign key relationships between:
  - Categories → Books
  - Members → Loans
  - Books → Loans
- Verified referential integrity to prevent orphan records.

---

## 4 Application Architecture

The application follows a **modular WinForms architecture**:

- **MainForm**
  Acts as the central navigation hub.
- **CategoryForm**
  Manages book categories (Add, Update, Delete, View).
- **BookForm**
  Manages books and assigns them to categories.
- **MemberForm**
  Manages library members.
- **LoanForm**
  Handles book borrowing and return operations.

A shared **DBConnection class** is used to manage database connections across all forms.

---

## 5 Database Connectivity

A centralized database connection approach was implemented using a dedicated class:

DBConnection.GetConnection()

This approach:

- Avoids duplicated connection strings
- Ensures consistency across all forms
- Simplifies maintenance and debugging

---

## 6 User Interface Design

- Designed using **Windows Forms Designer**
- Each form contains:
    - TextBoxes for input
    - Buttons for CRUD operations
    - DataGridView for data display
- Event-driven programming used for user interactions

---

## 7 Error Handling & Debugging

During development, several issues were encountered and resolved:

### ✔ Visual Studio Failed Installation

- Visual Studio installation failed multiple times due to:
    - Corrupted workloads
    - .NET SDK conflicts
- Issue was resolved by:
    - Repairing Visual Studio
    - Reinstalling required workloads
    - Clearing NuGet cache

✔ **Dependency & Package Issues**

- Missing NuGet packages were resolved
- Redundant references were removed
- Build errors caused by version mismatch were fixed

✔ Fixed SQL Server schema incompatibility with MySQL

✔ Fixed IDENTITY, NVARCHAR, BIT incompatibilities

✔ Ensured foreign key integrity using InnoDB

---

## 8 Testing & Validation

- Each form was tested individually
- CRUD operations verified using live database data
- UI events validated through manual testing
- Data consistency verified after insert/update/delete operations

---

## 9 Conclusion

The **Library Management System** successfully delivers a functional, user-friendly desktop application with reliable database integration.
All major technical challenges—including database migration, dependency conflicts, and installation failures—were resolved during development.

The project demonstrates:

- Strong understanding of WinForms
- Proper database handling using ADO.NET
- Effective debugging and problem-solving skills