Office of the College of Informatics and Computing Sciences

Advanced Computer Programming

(IT-222)

BookNest: Digital Library Services

Jake V. Laylo

Jerome Maru Padre

Renan S. Leynes

BSIT-2105

Office of the College of Informatics and Computing Sciences

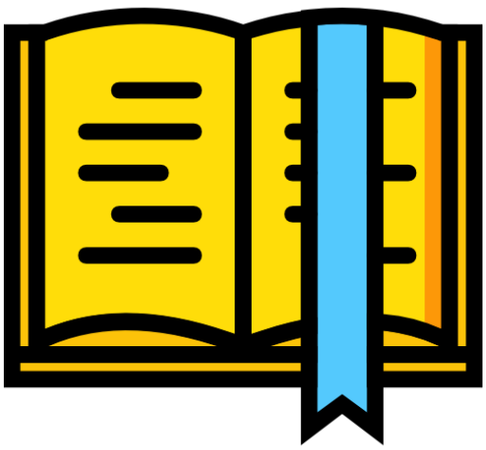
BookNest: Digital Library Services

In today's digital age, libraries have become more than just book collections; they are now dynamic knowledge hubs. To ensure smooth access to resources, improve user experiences, and streamline administrative tasks, a Library Management System (LMS) is crucial. This documentation is your all-in-one guide to grasp, implement, and make the most of our proposed Library Management System.

Vision and Mission:

BookNest envisions a future where access to knowledge is not bound by physical constraints. Our mission is to revolutionize the traditional concept of libraries, embracing the limitless possibilities of the digital realm. BookNest is more than a service; it's a commitment to making the wealth of human knowledge effortlessly accessible to all.

LOGO:



Statement of the Problem: The traditional library management system is grappling with several challenges that hinder its ability to meet the evolving needs of users and adapt to the digital age. The key issues include:

- **Manual Processes:** Many tasks, such as returning of books, borrowing, and cataloging, are still performed manually. This not only consumes a significant amount of time but also increases the likelihood of errors.
- **Manual Record-keeping:** Traditional libraries heavily rely on manual methods for cataloging, tracking, and managing books and other resources. This manual approach is time-consuming, prone to errors, and often results in outdated records.

- **Inefficient Borrowing and Returning Processes:** The process of borrowing and returning books is often cumbersome and time-intensive. Users may face long queues, delays, and difficulties in locating and retrieving books from the shelves.

Proposed Solutions:

To address the challenges inherent in traditional library management systems, the following solutions are proposed:

- **Implementation of Library Management System:** Introduce a comprehensive Library Management System (LMS) that automates cataloging, tracking, and management processes. This system should enable efficient data entry, reducing errors and improving the accuracy of library records.
- **Online Catalog and Digital Resources:** Develop an online catalog accessible to users, facilitating remote access to digital resources. This ensures that patrons can search and access materials from the comfort of their homes or any location with internet connectivity.

- **Automation of Borrowing and Returning Processes:** Implement self-service to automate the borrowing and returning of books that will be done through online transaction. This not only reduces wait times but also enhances the overall efficiency of library operations.

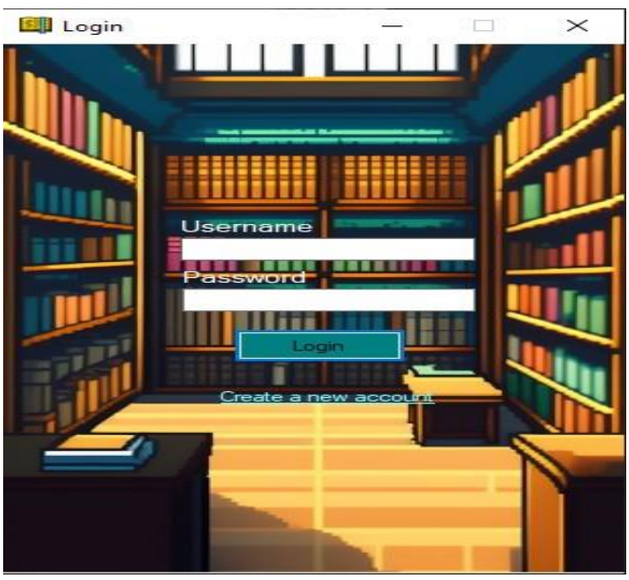
Conclusion:

BookNest: Digital Library Services is set to change how we see and use libraries, breaking free from physical constraints and embracing the endless opportunities in the digital world. This documentation is your key to grasp, apply, and leverage the potential of BookNest, ushering in a new era for library services in the digital age.

Here's the sample output of our system:

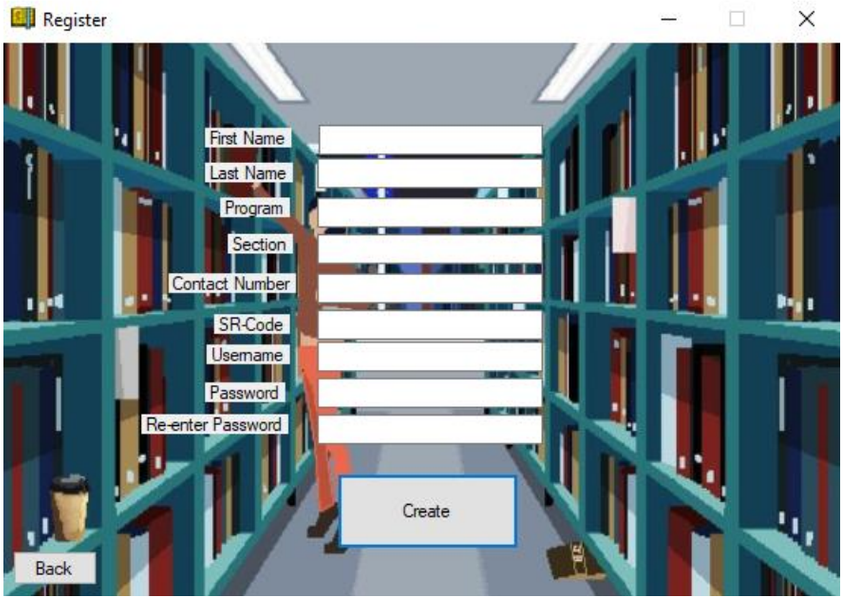
LOGIN MENU:

The login page is the gateway to the Library Management System, serving as the first point of interaction for both administrators and users.



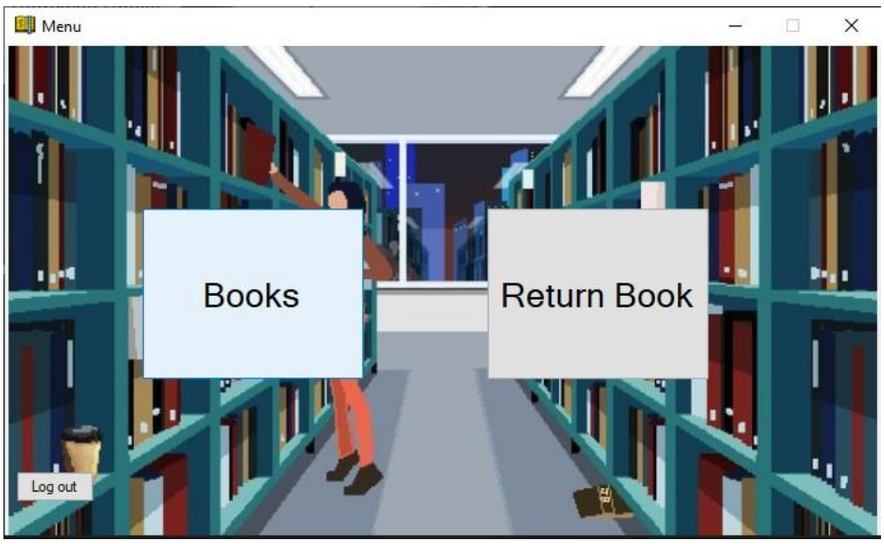
CREATE ACCOUNT: REGISTER STUDENTS

The "Register Students" feature within the Admin Menu of our Library Management System is a pivotal tool for administrators to facilitate seamless integration of students into the library community. This functionality simplifies the student registration process, ensuring a smooth onboarding experience and efficient access to library resources.



STUDENTS MENU:

The Students Menu in our Library Management System is a dedicated space for students to seamlessly interact with the library's resources. This menu serves two primary functions: viewing available books and facilitating the return of borrowed items.

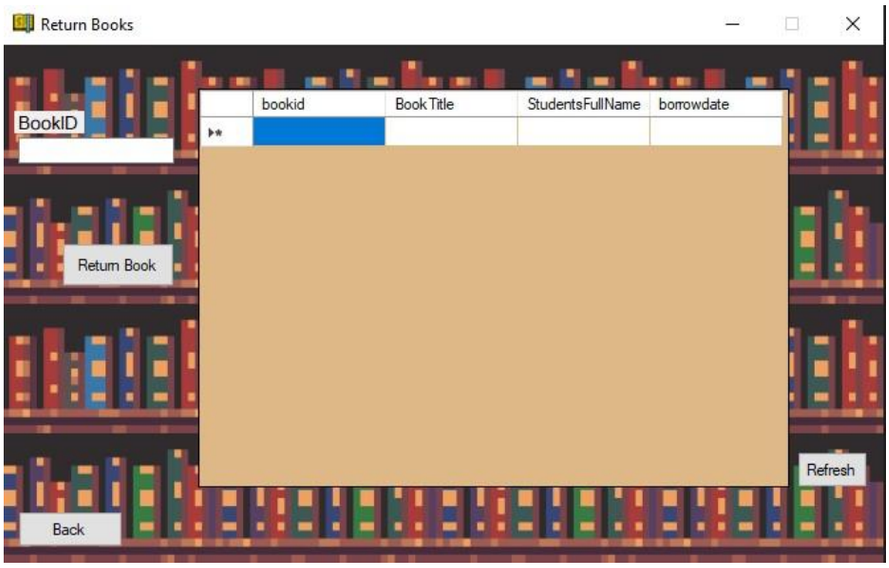


STUDENTS MENU: VIEWING AVAILABLE BOOKS TO BORROW

The "Viewing Available Books to Borrow" feature in our Library Management System is a gateway for students to access the library's resources. This feature enables students to effortlessly explore the available books, make informed choices, and seamlessly initiate the borrowing process.

STUDENTS MENU: RETURNING OF BOOKS

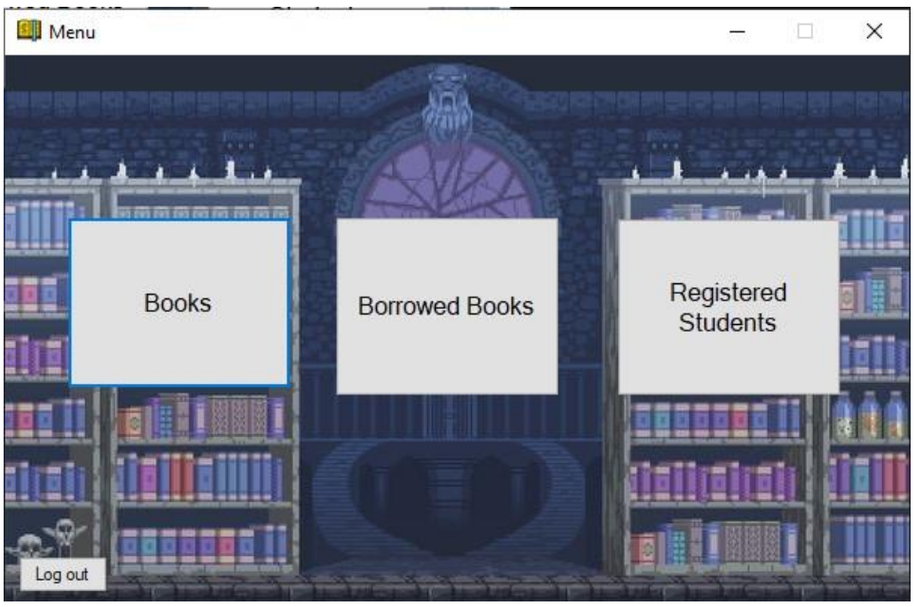
The "Returning of Books Students" feature is like the final chapter of borrowing books in our Library Management System. It helps students easily give back the books they borrowed, making sure everything happens smoothly. This way, we keep the books moving, and our library stays organized and ready for everyone to use.



ADMIN MENU:

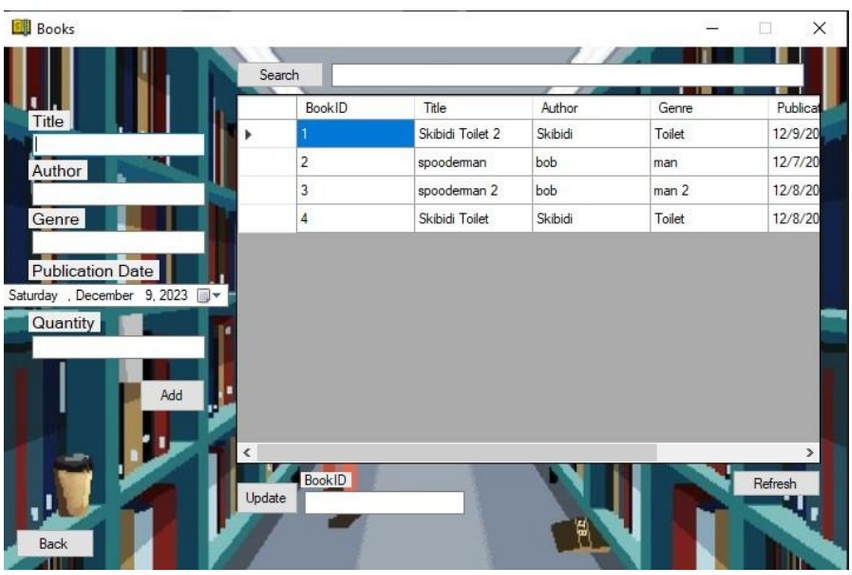
The Admin Menu in our Library Management System is the control center, where administrators wield the tools to effortlessly manage and optimize library operations. This menu empowers administrators with a range of features, including updating and adding books, viewing borrowed books, and managing registered students.

Office of the College of Informatics and Computing Sciences



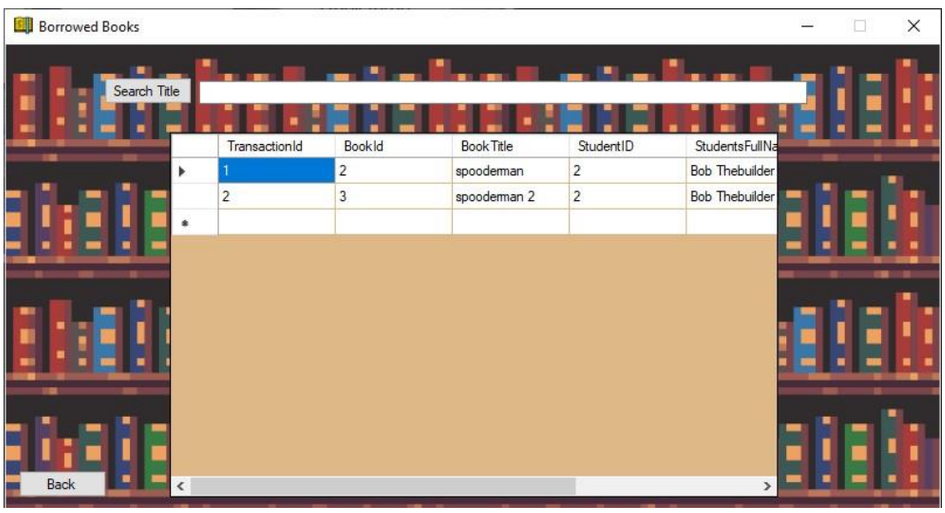
ADMIN MENU: UPDATING AND ADDING BOOKS

Within the Admin Menu of our Library Management System, administrators have the power to effortlessly update and books in the library's collection. This feature streamlines the process of keeping the catalog current, ensuring that library resources are accurately represented and readily accessible to users.



ADMIN MENU: VIEWING BORROWED BOOKS

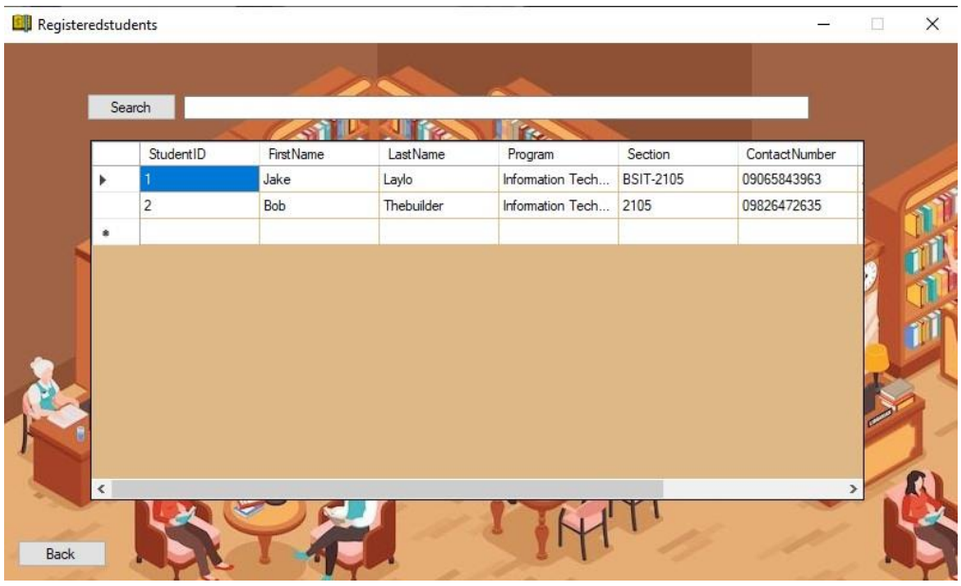
The "Viewing Borrowed Books" feature within the Admin Menu of our Library Management System equips administrators with a comprehensive tool to monitor and manage the current status of borrowed books. This functionality ensures efficient resource circulation and timely actions to enhance the overall library experience.



ADMIN MENU: VIEWING REGISTERED STUDENTS

The "Viewing Registered Students" feature within the Admin Menu of our Library Management System is a fundamental tool for administrators to manage and maintain an organized user database. This functionality ensures that administrators can easily oversee and engage with the student community, fostering a seamless and efficient library experience.

Office of the College of Informatics and Computing Sciences



Here's some sample code used in our System:

In our Library Management System, we employ robust password encryption techniques to safeguard user accounts and enhance the overall security posture of our system.

```
2 references
private string HashPassword(string password)
{
    using (SHA256 sha256 = SHA256.Create())
    {
        byte[] hashedBytes = sha256.ComputeHash(Encoding.UTF8.GetBytes(password));

        // Convert the hashed bytes to a hexadecimal string
        StringBuilder builder = new StringBuilder();
        for (int i = 0; i < hashedBytes.Length; i++)
        {
            builder.Append(hashedBytes[i].ToString("x2"));
        }

        return builder.ToString();
    }
}

1 reference
private bool IsStudent(string username, string password, out bool isLoggedIn)
{
    // Hash the entered password using SHA256
    string hashedPassword = HashPassword(password);

    // Check student credentials in the database
    using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=librarymanagementsystem"))
    {
        connection.Open();
        string query = "SELECT COUNT(*) FROM students WHERE Username = @Username AND Password = @Password";
        using (MySqlCommand command = new MySqlCommand(query, connection))
        {
            command.Parameters.AddWithValue("@Username", username);
            command.Parameters.AddWithValue("@Password", hashedPassword);
            int count = Convert.ToInt32(command.ExecuteScalar());

            // Check if the user is logged in and update the database accordingly
            if (count > 0)
            {
                isLoggedIn = true;
                UpdateLoggedInStatus(username, true);
            }
            else
            {
                isLoggedIn = false;
            }

            return count > 0;
        }
    }
}
```

Office of the College of Informatics and Computing Sciences

In our Library Management System, the process of inserting a book query code involves integrating a segment of code into the system that facilitates efficient retrieval and management of information related to books.

```
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=librarymanagementsystem"))
        {
            connection.Open();


            string title = textBox2.Text;
            string author = textBox3.Text;
            string genre = textBox4.Text;
            DateTime publicationDate = Convert.ToDateTime(dateTimePicker1.Text);
            int quantity = Convert.ToInt32(textBox6.Text);

            // Insert or update the book in the books table using the composite key (Title, Author)
            string insertOrUpdateQuery = "INSERT INTO books (Title, Author, Genre, PublicationDate, Quantity) " +
                                         "VALUES (@Title, @Author, @Genre, @PublicationDate, @Quantity) " +
                                         "ON DUPLICATE KEY UPDATE Quantity = Quantity + @Quantity";

            using (MySqlCommand command = new MySqlCommand(insertOrUpdateQuery, connection))
            {
                command.Parameters.AddWithValue("@Title", title);
                command.Parameters.AddWithValue("@Author", author);
                command.Parameters.AddWithValue("@Genre", genre);
                command.Parameters.AddWithValue("@PublicationDate", publicationDate);
                command.Parameters.AddWithValue("@Quantity", quantity);

                int rowsAffected = command.ExecuteNonQuery();

                if (rowsAffected > 0)
                {
                    MessageBox.Show("Book added/updated successfully!", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
                else
                {
                    MessageBox.Show("Failed to add/update book.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
                }
            }
        }
```

Search functionality is a critical component of the Library Management System, enabling users to efficiently explore and retrieve relevant information from the extensive catalogue of resources.

```
1 reference
private void SearchBooks(string searchTerm)
{
    try
    {
        using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=librarymanagementsystem"))
        {
            connection.Open();

            string query = "SELECT * FROM Books WHERE BookID LIKE @SearchTerm OR Title LIKE @SearchTerm OR Author LIKE @SearchTerm OR Genre LIKE @SearchTerm OR PublicationDate LIKE @SearchTerm OR Quantity LIKE @SearchTerm";

            using (MySqlCommand command = new MySqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@SearchTerm", $"%{searchTerm}%");


                using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))
                {
                    DataTable searchResultTable = new DataTable();
                    adapter.Fill(searchResultTable);


                    if (searchResultTable.Rows.Count == 0)
                    {
                        MessageBox.Show("No books found with the specified search term.", "No Results", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }

                    // Bind the DataTable to the DataGridView
                    dataGridView1.DataSource = searchResultTable;
                }
            }
        }
    }
    catch (Exception ex)
    {

        MessageBox.Show($"Error: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
```

Office of the College of Informatics and Computing Sciences

In our Library Management System, the refresh functionality serves as a vital tool for users to update and synchronize the displayed information with the most recent data available within the system.

```
//refresh the gridview
1 reference
private void button5_Click(object sender, EventArgs e)
{
    LoadBooksData();
}
```

The Borrowed Books functionality in our Library Management System is a pivotal feature designed to manage and track the borrowing activities of users.

```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    try
    {
        using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=librarymanagementsystem"))
        {
            connection.Open();

            // Get student ID where loggedin is true
            int studentId;
            string getStudentIdQuery = "SELECT StudentId FROM students WHERE loggedIn = true";

            using (MySqlCommand getStudentIdCmd = new MySqlCommand(getStudentIdQuery, connection))
            {
                object result = getStudentIdCmd.ExecuteScalar();

                if (result != null && int.TryParse(result.ToString(), out studentId))
                {
                    // Get book ID from the textbox
                    int bookId;

                    if (int.TryParse(textBox1.Text, out bookId))
                    {
                        // Check book quantity
                        string checkQuantityQuery = "SELECT quantity FROM books WHERE BookId = @bookId";

                        using (MySqlCommand checkQuantityCmd = new MySqlCommand(checkQuantityQuery, connection))
                        {
                            checkQuantityCmd.Parameters.AddWithValue("@bookId", bookId);

                            object quantityResult = checkQuantityCmd.ExecuteScalar();
```

In our Library Management System, the "Insert Transaction on Returning Books" functionality plays a crucial role in maintaining an accurate record of borrowing and returning activities.

```
if (quantityResult != null && int.Parse(quantityResult.ToString()) > 0)
{
    // Insert borrow record
    string insertBorrowQuery = "INSERT INTO borrowedbooks (bookid, studentid, borrowdate, Borrowedqty) " +
                               "VALUES (@bookId, @studentId, @borrowDate, 1)";

    using (MySqlCommand insertBorrowCmd = new MySqlCommand(insertBorrowQuery, connection))
    {
        insertBorrowCmd.Parameters.AddWithValue("@bookId", bookId);
        insertBorrowCmd.Parameters.AddWithValue("@studentId", studentId);
        insertBorrowCmd.Parameters.AddWithValue("@borrowDate", DateTime.Now);

        insertBorrowCmd.ExecuteNonQuery();

        MessageBox.Show("Book borrowed successfully.", "Success", MessageBoxButtons.OK, MessageBoxIcon.Information);
    }

    // Update book quantity
    string updateQuantityQuery = "UPDATE books SET quantity = quantity - 1 WHERE bookid = @bookId";

    using (MySqlCommand updateQuantityCmd = new MySqlCommand(updateQuantityQuery, connection))
    {
        updateQuantityCmd.Parameters.AddWithValue("@bookId", bookId);

        updateQuantityCmd.ExecuteNonQuery();
    }
}
else
{
    MessageBox.Show("Book Unavailable", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);
}
```

The "Transaction Query for GridView" functionality in our Library Management System serves as a powerful tool for administrators to retrieve and display transaction data in a structured format.

```
1 reference
private void LoadTransactionsBooks()
{
    try
    {
        using (MySqlConnection connection = new MySqlConnection("server=localhost;port=3306;username=root;password=;database=librarymanagementsystem"))
        {
            connection.Open();


            string query = "SELECT transactions.TransactionId, " +
                           "transactions.BookId, " +
                           "books.title AS BookTitle, " +
                           "transactions.StudentID, " +
                           "CONCAT(students.firstname, ' ', students.lastname) AS StudentsFullName, " +
                           "transactions.BorrowDate, " +
                           "transactions.ReturnDate " +
                           "FROM transactions " +
                           "JOIN books ON transactions.bookid = books.bookid " +
                           "JOIN students ON transactions.studentid = students.studentid";

            using (MySqlCommand command = new MySqlCommand(query, connection))
            {
                using (MySqlDataAdapter adapter = new MySqlDataAdapter(command))
                {
                    DataTable borrowedbooksTable = new DataTable();
                    adapter.Fill(borrowedbooksTable);

                    // Bind the DataTable to the DataGridView
                    dataGridView1.DataSource = borrowedbooksTable;

                    // Make the DataGridView read-only
                    dataGridView1.ReadOnly = true;

                    // Make all columns read-only
                    foreach (DataGridViewColumn column in dataGridView1.Columns)
                    {
                        column.ReadOnly = true;
                    }
                }
            }
```